U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards and Technology

FIPS

FIPS PUB 190

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION

GUIDELINE FOR THE USE OF ADVANCED AUTHENTICATION TECHNOLOGY ALTERNATIVES

CATEGORY: COMPUTER SECURITY SUBCATEGORY: ACCESS CONTROL

1994 SEPTEMBER 28

FIPS PUB 190

FIPS PUB 190

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION

GUIDELINE FOR THE USE OF ADVANCED AUTHENTICATION TECHNOLOGY ALTERNATIVES

CATEGORY: COMPUTER SECURITY SUBCATEGORY: ACCESS CONTROL

Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-0001

Issued September 28, 1994

U.S. Department of Commerce
Ronald H. Brown, Secretary

Technology Administration
Mary L. Good, Under Secretary for Technology

National Institute of Standards
and Technology
Arati Prabhakar, Director

Foreword

The Federal Information Processing Standards Publication Series of the National Institute of Standards and Technology (NIST) is the official publication relating to standards and guidelines adopted and promulgated under the provisions of Section 111(d) of the Federal Property and Administrative Services Act of 1949 as amended by the Computer Security Act of 1987, Public Law 100-235. These mandates have given the Secretary of Commerce and NIST important responsibilities for improving the utilization and management of computer and related telecommunications systems in the Federal Government. The NIST, through its Computer Systems Laboratory, provides leadership, technical guidance, and coordination of Government efforts in the development of standards and guidelines in these areas.

Comments concerning Federal Information Processing Standards Publications are welcomed and should be addressed to the Director, Computer Systems Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899.

James H. Burrows, Director
Computer Systems Laboratory

Abstract

This Guideline describes the primary alternative methods for verifying the identities of computer system users, and provides recommendations to Federal agencies and departments for the acquisition and use of technology which supports these methods. Although the traditional approach to authentication relies primarily on passwords, it is clear that password-only authentication often fails to provide an adequate level of protection. Stronger authentication techniques become increasingly more important as information processing evolves toward an open systems environment. Modern technology has produced authentication tokens and biometric devices which are reliable, practical, and cost-effective. Passwords, tokens, and biometrics can be used in various combinations to provide far greater assurance in the authentication process than can be attained with passwords alone.

Key words: computer security; cryptographic modules; cryptography; Federal Information Processing Standards Publication; telecommunication security.

National Institute of Standards
and Technology
FIPS PUB 190
62 pages (Sept. 28, 1994)
CODEN: FIPPAT

U.S. Government Printing Office
Washington: 1994

For sale by the National
Technical Information
Service
U.S. Department of Commerce
Springfield, VA 22161

Federal Information Processing Standards Publication 190

1994 September 28

ANNOUNCING THE

GUIDELINE FOR THE USE OF ADVANCED AUTHENTICATION TECHNOLOGY ALTERNATIVES

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology (NIST) after approval by the Secretary of Commerce pursuant to Section 111(d) of the Federal Property and Administrative Services Act of 1949 as amended by the Computer Security Act of 1987, Public Law 100-235.

1. **Name of Guideline.** Guideline For The Use Of Advanced Authentication Technology Alternatives (FIPS PUB 190).

2. **Category of Guideline.** Computer Security, **Subcategory** Access Control.

3. **Explanation.** This Guideline describes the primary alternative methods for verifying the identities of computer system users, and provides recommendations to Federal agencies and departments for the acquisition and use of technology which supports these methods. Although the traditional approach to authentication relies primarily on passwords, it is clear that password-only authentication often fails to provide an adequate level of protection. Stronger authentication techniques become increasingly more important as information processing evolves toward an open systems environment. Modern technology has produced authentication tokens and biometric devices which are reliable, practical, and cost-effective. Passwords, tokens, and biometrics can be used in various combinations to provide far greater assurance in the authentication process than can be attained with passwords alone.

4. **Approving Authority.** Secretary of Commerce.

5. **Maintenance Agency.** Department of Commerce, National Institute of Standards and Technology Computer Systems Laboratory.

6. **Cross Index.**

   a. FIPS PUB 46-2, Data Encryption Standard.

   b. FIPS PUB 48, Guidelines on Evaluation of Techniques for Automated Personal Identification.

   c. FIPS PUB 74, Guidelines for Implementing and Using the NBS Data Encryption Standard.

   d. FIPS PUB 81, DES Modes of Operation.

   e. FIPS PUB 83, Guideline of User Authentication Techniques for Computer Network Access Control.

   f. FIPS PUB 112, Password Usage.

   g. FIPS PUB 113, Computer Data Authentication.

h. FIPS PUB 171, Key Management Using ANSI X9.17.

i. FIPS PUB 180, Secure Hash Standard.

j. Special Publication 500-157, Smart Card Technology: New Methods for Computer Access Control.

k. Special Publication 800-2, Public Key Cryptography.

Other NIST publications may be applicable to the use of this guideline. A list (NIST Publications List 91) of currently available computer security publications, including ordering information, can be obtained from NIST.

7. **Applicability.** This guideline is applicable to all Federal departments and agencies that use authentication systems to protect unclassified information within computer and telecommunication systems (including voice systems) that are not subject to Section 2315 of Title 10, U.S. Code, or Section 3502(2) of Title 44, U.S. Code. This guideline may be used by all Federal departments and agencies in designing, acquiring and implementing authentication systems within computer and telecommunication systems (including voice systems) that they operate or that are operated for them under contract. Non-Federal government organizations are encouraged to use this guideline when it provides the desired security for protecting valuable or sensitive information.

8. **Applications.** Authentication systems may be utilized in various computer and telecommunication (including voice) applications and in various environments (e.g., centralized computer facilities, office environments, hostile environments). The strength of an authentication system should be chosen to provide a degree of assurance appropriate for the security requirements of the application and environment in which the system is to be utilized and the security services which the system is to provide.

9. **Specifications.** Federal Information Processing Standards (FIPS) Guideline 190, Guideline For The Use Of Advanced Authentication Technology Alternatives (affixed).

10. **Export Control.** Many of the authentication systems discussed in this guideline make use of cryptographic techniques to strengthen the security of the authentication process. Certain cryptographic devices and technical data regarding them are deemed to be defense articles (i.e., inherently military in character) and are subject to Federal government export controls as specified in Title 22, Code of Federal Regulations, Parts 120-128. Some exports of cryptographic systems and technical data regarding them must comply with these Federal regulations and be licensed by the U.S. Department of State. Other exports of cryptographic systems and technical data regarding them fall under the licensing authority of the Bureau of Export Administration of the U.S. Department of Commerce. The Department of Commerce is responsible for licensing cryptographic devices used for authentication, access control, proprietary software, automatic teller machines (ATMs), and certain devices used in other equipment and software. For advice concerning which agency has licensing authority for a particular cryptographic device, please contact the respective agencies.

11. **Implementation Schedule.** This guideline becomes effective May 1, 1995.

12. **Qualifications.** The authentication technology described in this guideline is based upon information provided by many sources within the Federal government and private industry. Authentication systems are designed to protect against adversaries mounting cost-effective attacks on unclassified government or commercial data (e.g., hackers, organized crime, economic competitors). The primary goal in designing an effective security system is to make the cost of any attack greater than the possible payoff.

13. **Where to obtain copies.** Copies of this publication are available for sale by the National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. When ordering, refer to Federal Information Processing Standards Publication 190 (FIPSPUB190), and title. When microfiche is desired, this should be specified. Payment may be made by check, money order, credit card, or deposit account.

Federal Information
Processing Standards Publication 190

1994 September 28

Specifications for

GUIDELINE FOR THE USE OF ADVANCED AUTHENTICATION TECHNOLOGY

CONTENTS

CONTENTS (continued)

1. INTRODUCTION

This Guideline provides information and guidance to Federal agencies on the use of advanced authentication technology as a critical element in the design of effective access control mechanisms for automated systems which process unclassified information. As the trend toward networking continues, the ability to verify the identity of system users with a high degree of accuracy becomes more important. Systems which cannot differentiate between requests for service by legitimate users and unauthorized access attempts are vulnerable to a variety of attacks. Although passwords are the traditional method for verifying the identity of users, there are several alternative methods which can enhance the security of an access control system. This document describes these methods and provides recommendations for their use. Each major section contains an example authentication system based upon the technology described in that particular section. The examples are constructed specifically for the purposes of this document, with the exception of the Advanced Smartcard Access Control System presented in Section 4. However, all examples are based on technology that is available now or is expected in the near future. Discussion of specific commercial products does not constitute an endorsement by NIST.

2. PRINCIPLES OF AUTHENTICATION

The broadest definition of authentication within computing systems encompasses identity verification, message origin authentication, and message content authentication [1]. The concept of identity verification specifically applies to principals with information processing and decision making capabilities, including human users, computing systems and processes executing on those systems. From an authentication standpoint, the term "user" applies to all these principals. This Guideline focuses on technology and techniques for verifying the identity of human users, but many of these techniques are equally applicable to authentication of other principal types. Authentication through knowledge of secret information or possession of a unique physical authentication token are equally valid for all the types of entities described above. On the other hand, biometric authentication only makes sense in the context of human users.

Reliable authentication mechanisms are critical to the security of any automated information system. If the identity of legitimate users can be verified with an acceptable degree of accuracy, those attempting to gain access without proper authorization can be denied permission to use the system. When a legitimate user's identity is verified, access control techniques are applied to mediate that user's access to system resources. If a computer system cannot verify the identity of users and other computers, the system will not be able to protect itself against unauthorized access. A variety of methods are available for performing user authentication, and these methods form the basis for access control systems [2]. The three generally accepted categories of methods for verifying the identity of a user are based on something the user KNOWS, such as a password; something the user POSSESSES, such as an authentication token; and some PHYSICAL CHARACTERISTIC of the user, such as a fingerprint or voice pattern. In order to use these characteristics to verify the identity of an individual, computer systems use software, hardware, or a combination of both.

In the past, it was relatively easy to protect computer systems because they were typically installed in a centralized computing facility. Since the terminals used to access the computer were usually in the same building, only those persons having physical access to the building would be able to use the terminals. With the proliferation of networked computer systems, however, this level of physical access control is no longer viable. The design of open computing systems permits access to more systems, and some of these access attempts may not be by legitimate users. Users may be able to access network-connected computers from any physical location on the network, and the logical connection which supports a session between the user and a given computer may travel through many communications circuits. The increasing level of interconnection between computer systems has made it possible to distribute and process information far more easily than in the past. However, it has also become significantly more difficult to identify system users based on physical location, since the pathway between a user and the computing resources accessed by that user may be impossible to trace.

Attackers often take advantage of the anonymity provided by communications networks when attempting to break into a target machine. A significant amount of effort is usually required to locate and prosecute these attackers, primarily because of the difficulty of tracing an attacker's access routes through communications networks which may span international boundaries.

Networking not only makes it more difficult to identify system users, it also increases the opportunities for unauthorized parties to intercept authentication data passing through the network during the course of a legitimate session between a user and a remote host computer. User passwords are sometimes transmitted through a network in plaintext form. If an attacker is able to monitor the user's session, the attacker may be able to record the user's password or other critical authentication data. This would allow the attacker to pose as a valid user by initiating a login on the remote host and submitting the user's authentication data when the host requests it. Software is readily available for monitoring network traffic, primarily for the purpose of performance management and problem diagnosis. Unfortunately, the same software is often quite effective at capturing passwords as they are transmitted through a network.

Some systems apply a cryptographic algorithm to scramble (encrypt) passwords before they are transmitted, so that the plaintext password is not exposed. However, an attacker may still be able to record the encrypted password, and gain access to the host computer by submitting the encrypted value. In either case, the host computer will be unable to distinguish between the attacker and a valid user, and will grant access to the attacker.

In a modern automated information system, processes running on one computer may interact with other computers in order to transfer information or access common resources. These interactions may take place across networks and involve machines which are not located in the same facility. For example, many electronic mail protocols require the transfer and routing of information through computers which are heterogeneous in terms of ownership and physical location. It is therefore necessary to consider situations where one computer needs to verify the identity of another computer, with or without intervention from a human user. It is usually desirable in these cases to implement some form of mutual authentication, whereby the identity of each computer is verified simultaneously. Fortunately, computers are capable of implementing cryptographic authentication protocols which provide an efficient and secure means for performing mutual authentication (Section 7).

Human users often access multiple services on multiple host computers in modern automated information systems. Separate authentication events may be required for each service a user wishes to access, particularly if these services are resident on separate host machines. Users might, for example, be required to demonstrate possession of a physical authentication token for each service. In some cases, services or host computers may even use different authentication techniques which would, for example, force users to memorize passwords for some services and carry tokens or provide biometric scans for others. This situation quickly becomes an unreasonable burden for users, and can lead to poor security practices.

To address the problems described above, logon authentication schemes have been developed that only require users to authenticate once during a session. These approaches are commonly referred to as unitary logon or single sign-on. Unitary logon is generally a two-step process, in which the user first authenticates to a principal. The principal may be the user's workstation, a physical authentication token, or some other device. Then, as the user requests access to various services, the principal is responsible for authenticating the user to each service. Conceptually, the principal acts as a proxy for the user in conveying the original authentication event and automates subsequent authentications with little or no intervention from the user. These subsequent authentications are usually based on strong cryptographic protocols which are secure across communications networks. It should be noted that each service accessed by a user must understand the protocol for interacting with the principal responsible for authenticating the user. Also, the principal must be responsible for determining the point at which a given user's current authentication terminates. This termination point is often tied to the end of a user's login session.

3. PASSWORD BASED AUTHENTICATION

3.1 Overview

The traditional method for authenticating users has been to provide them with a secret password, which they must submit when requesting access to a particular system. The majority of computer systems in use today rely on passwords for authentication. The primary advantage of password-only authentication is that it can be implemented entirely in software, thus avoiding the cost of special purpose authentication hardware. However, password systems have a number of disadvantages in practice which restrict their use to applications with minimal security requirements, or situations where password management can be strictly controlled. Password based authentication is most effective when combined with other authentication techniques.

3.2 Factors Affecting Password Security

Passwords may be chosen as the sole means of authentication, or may be combined with other authentication methods for improved security. A number of factors affect the security of a system which relies on passwords for authentication. These factors include the composition, length, lifetime, source, ownership, distribution, storage, entry, transmission, and authentication period of the passwords. Federal Information Processing Standards Publication 112 [3] describes these factors in detail, and so they will be discussed only briefly in this document.

3.2.1 Composition

The composition of a password refers to the range of values from which each character of the password may be chosen. For example, a particular implementation might allow each character of a password to be chosen from the set of letters in the alphabet. This would yield 26 possible values for each character, assuming case insensitivity. For the purposes of this example, assume that the host system allocates eight bits, or one byte, of storage for each character. One byte can represent any of 256 possible values, which is approximately ten times the number of letters in the alphabet. By restricting the range of possible values for each character to the 26 letters of the alphabet, the security of the password system is decreased. Exhaustive attacks involve the submission of as many different password values as possible in the hopes of finding one or more which are valid. The work factor for someone attempting an exhaustive attack is directly related to the number of possible values which must be tried for each character of the password. However, it is often necessary to restrict the range of allowable values for practical reasons. Many keyboards do not allow the user to enter all possible values for a character. Numeric keypads are often used for the entry of Personal Identification Numbers (PINs), which are passwords composed only of numeric characters. These keypads are typically found in automated teller machines used by the banking industry, but are also used in a variety of other access control applications. A numeric keypad usually allows for the entry of decimal digits 0 through 9, thus restricting the range of each character of a PIN to ten possibilities.

It may also be necessary to restrict the range of allowable password characters for mnemonic reasons. If password characters are chosen at random from the full range of possible values, users will find it difficult to remember these passwords. Random combinations of characters are difficult to remember since human users will interpret many of them as nonsense. In such cases, users are much more likely to write passwords down because they cannot be memorized easily. Automated systems may use a password generator which produces pronounceable non-word combinations of characters. For example, passwords produced by this type of system might be of the form consonant- vowel- consonant- consonant- vowel- consonant, excluding words which appear in a dictionary. This approach eliminates the threat of dictionary attacks, where words are chosen in sequence from a dictionary for submission as passwords. Users should be able to remember pronounceable non-words more easily than totally random combinations of characters, reducing the likelihood that passwords will be written down. Password generation schemes are often a compromise between the security of random password generation and the need to produce passwords which users can remember.

3.2.2 Length

The length of a password refers to the total number of characters which make up the password. In combination with the range of values allowed for each character, the length determines the total number of possible password values. A password system which uses the decimal digits zero through nine with a length of four would have a range of ten to the fourth power, or ten thousand possible password values. As the length and/or composition parameters are increased, the number of possible password values increases proportionally. Increasing these parameters should have a positive effect on the overall security of the system, since exhaustive attacks become more difficult. However, system users will have more trouble remembering their passwords as the length and composition are increased.

3.2.3 Lifetime

If user passwords are not changed at reasonable intervals, it becomes more likely that passwords could be compromised by exhaustive search techniques. The lifetime of a password determines the amount of time which an attacker can use to attempt to compromise the password through exhaustive search or other techniques. If an attacker manages to guess a password which has been replaced with a new password, the attacker has gained nothing.

This scenario assumes that the new password value bears no relationship to the old password, as would be the case if new passwords were generated randomly. In cases where users are allowed to choose their own passwords, however, they frequently choose values which are a variation on old password values. For example, a user may choose the password "bbcdef" if the user's previous password was "abcdef". The new password is easier to remember, since it only differs from the old password by one letter. This situation increases the risk that an attacker could guess the new password value, since knowing the old password would provide some information about the possible values of the new password.

The password lifetime chosen for an application should balance the apparent security of a short lifetime against the burden placed on users when passwords are changed too often. Users may become frustrated when required to constantly change and memorize new passwords, making it more likely that trivial passwords will be chosen.

3.2.4 Source

The source which generates new passwords in a system has a major impact on the security of that system. If passwords are generated by an automated system, that system component will be responsible for ensuring the security of password values. Automated password generators will, by definition, know the value of each new password in the system. Care must be taken in the design and operation of password generators to ensure that they can be trusted, since an access control system would be rendered useless if the password generation process were not secure. NIST has developed a standard for automated password generation [4].

Users may be allowed to choose their own passwords, rather than having them chosen by an automated system. In these situations, the passwords chosen by users should be checked by automated means to ensure that weak passwords are rejected. For example, the security policy of a system might set the following requirements: user-chosen passwords must be at least six characters in length, they must not appear in a dictionary of English words, and they must differ from the user's previous password by at least two characters. Any user-chosen passwords not meeting these requirements would be rejected and the user would be asked to choose another password.

3.2.5 Distribution

Passwords which are generated automatically must be distributed to system users. The communications lines which carry new passwords from the host system to users should be protected from attempts to intercept passwords. This can be difficult when passwords must travel through networks which span organizational and geographic boundaries. Encryption can be used to scramble passwords which must travel through unprotected networks, so that they become unintelligible to an attacker. In the case where users choose their own passwords, the passwords must be sent to the host system after they have been selected by the users.

Whether passwords are distributed in hardcopy form, electronically, or through other means, the distribution process should provide protection against disclosure. Sealed envelopes with tamper-evident features are often used for distribution of hardcopy passwords. If an unauthorized party intercepts a tamper-evident envelope and opens it to read the password, the envelope cannot be resealed and sent to the intended recipient without evidence of tampering. This approach relies on the system users to recognize and report suspected disclosure of hardcopy passwords. If a password is compromised in this fashion, there may be a short period of time before the legitimate user detects and reports the compromise. An attacker may be able to use the password to gain access to the system during this time, because the password is considered valid until the user reports that it has been compromised.

3.2.6 Storage

In addition to the generation and distribution of passwords, a system must store passwords for use in the authentication process. When a user attempts to login to the system, the user will submit a password which must be compared to the stored password, or some one-way mapping thereof, which the system knows to be valid for that user. Protection can be provided for passwords by storing them in a physically separate area which can only be accessed by authorized system components. Stored passwords may also be protected by encryption or through the application of a one-way mapping function before storage. Data encryption is described in Section 7.1.

3.2.7 Entry and Transmission

Users must submit passwords to the host system during a login, and possibly at other times during a normal session. A user's password may be subject to disclosure while the user is entering the password. The terminal should not display the password as the user enters it, so that others cannot read the password from the user's display. Users should be allowed more than one attempt to enter a password during a login, since the user may accidently mistype the password. However, there should be a limit to the number of incorrect password entry attempts to protect against exhaustive search attacks, as described in Section 3.2.1. Many systems allow three password entry attempts before locking a user out. The user is then required to notify a system administrator or security officer in order to obtain a new password.

After the password has been entered, the user's terminal transmits it to the host system unless the user is accessing the host via a main system console. As the password travels from the user's terminal to the host, it is subject to disclosure if the line between the terminal and the host is not secure. The risk of exposure during transmission of the password from the user's terminal increases as a function of the complexity of the network which connects the terminal to the host. Networks vary in complexity depending on the number of access points, the number of sessions which can be carried simultaneously, the degree of physical protection provided for data on the network, and a variety of other factors. Encryption of passwords prior to transmission or the use of a cryptographic authentication protocol which does not rely on transmission of plaintext passwords can reduce or eliminate this risk. However, encryption alone does not protect against replay because an attacker may be able to record the encrypted password and play it back in encrypted form to gain access. Inclusion of a time variant parameter in the encrypted password message can protect against replay attacks.

3.3 Problems with Password-Only Authentication

Policies and procedures have been developed for the management of password-only authentication techniques. However, these techniques are sometimes difficult to implement effectively in real-world situations. Some of the factors which influence the security of a password system may be beyond the control of those responsible for managing the system. During the development of a computer system, it is common practice for the system developers to use master passwords which provide total control over the system for debugging purposes. These passwords are sometimes left in the product, either inadvertently or intentionally, when the system goes into production. When this is done intentionally, it provides the developer with a convenient "back door" entry into the customer's system which facilitates product support and maintenance. However, this is a dangerous practice because an intruder may be able to gain complete control over the system by learning the developer's password. In addition, the customer may not wish to trust the manufacturer with this level of control over a system after it is installed at the customer's site. Customers should verify that passwords used by the manufacturer during system development and installation have been removed before the system is used.

The password problem is multiplied when users access remote computing resources through a network. Because it is difficult to control physical access to remote terminals, it is possible for an attacker to make repeated attempts to guess passwords on host computers connected to the network. In addition, passwords are often transmitted to a remote computer to authenticate the user. Transmitting static passwords over a network in plaintext form can drastically increase the opportunities for an attacker to capture them directly from the communications line, or from a computer which is acting as an intermediate node in the transmission process. There have been numerous well-publicized cases of intruders breaking into computer systems by guessing or stealing passwords.

Authentication which relies solely on passwords has often failed to provide adequate protection for computer systems for a number of reasons. If users are allowed to make up their own passwords, they tend to choose ones which are easy to remember, and therefore easy to guess. If passwords are generated from a random combination of characters, users often write them down because they are difficult to remember. Systems which use only passwords for authentication should provide strong mechanisms for controlling the generation, distribution, and use of system passwords. Password systems can be effective if managed properly, but this is seldom the case. Advances in security technology provide a number of alternative authentication methods which can be used alone or in combination with passwords to improve the security of an access control system.

3.4 Example

A hypothetical system will be used to illustrate the application of good password management techniques in an access control system. This system consists of a number of host computers, or servers, interconnected by a local area network. Users access the services provided by the host computers through intelligent workstations which may in some cases also serve as hosts for other users. Only unclassified information is stored on and processed by the system. A security officer is assigned for each host, and in most cases also plays the role of system administrator for that machine. Host systems rely entirely on passwords to verify the identity of users requesting services. This scenario is typical of many networked computer systems.

The security policy for all host computers in this hypothetical network dictates certain rules for the generation, distribution, and management of user passwords. Some of the processes required by the security policy involve cryptographic techniques which are described more fully in Section 7. The requirement for cryptographic protection of passwords as they pass through the network increases overall security. However, the use of cryptography also complicates the authentication architecture. In particular, protocols for the generation, distribution, and management of cryptographic keys must be included. Certain aspects of the security policy are enforced by the operating system or special applications programs executing on the host systems. For example, password length and composition are checked automatically each time a user's password is changed. This check is performed by the same software which is responsible for managing changes to the password database. A simple set of rules for password management in this system follows:

1. Passwords are composed of the characters available on a standard

computer keyboard, i.e., letters of the alphabet, numeric digits, and punctuation. When passwords are created, the system performs a series of checks to make sure that the passwords chosen are not weak.

2. Passwords are at least six characters in length.

3. Passwords must be changed every four months. Users are notified by the system when individual passwords have reached the four month expiration date. The system prompts individual users for new passwords, and does not allow further access until a user's password has been properly updated.

4. Passwords are distributed to users through personal interaction with security officers, or through delivery by a trusted courier in a sealed tamper- evident envelope. Passwords are never distributed through routine interoffice mail services.

5. Passwords are stored on host systems for comparison purposes. Before storage, passwords are scrambled using a one-way mapping algorithm to provide protection for the stored values. The original password values cannot be recovered from the scrambled values, so when a user submits a password for authentication purposes the system must one-way map the password and compare the result to the scrambled value originally stored for that user. Even if the stored value is compromised, the plaintext password must still be derived by exhaustive search.

6. When a user wishes to access services on a host system, the user must submit a password. The password is entered at the user's workstation, and must often be transmitted to the host system via the local area network. While the user is typing the password, the workstation does not echo it to the display. The workstation then encrypts the password and a time-variant parameter, and transmits the result to the host system. The host system decrypts the password, recovering the original form of the password entered by the user. The one-way mapping algorithm is then applied to encrypt the password, and this encrypted form is compared to the encrypted password value in the password database for this user. Encryption provides protection for the password as it is transmitted through the network from the user's workstation to the host system.

7. Users are not allowed to write down passwords, or to share them with other users. Users are made aware of this requirement before they are given access to the system, and are also made aware of any corrective actions which will be taken if this rule is violated.

The requirement for encryption in item 6 contributes to the security of the system, because passwords are not exposed in plaintext form during transmission. The system design must include workstations which have cryptographic capability, and a protocol for managing the cryptographic keys which must be shared between workstations and host computers. In addition, the generation and verification of time-variant parameters requires time synchronization between appropriate system components. These requirements complicate the system design to a certain extent, but the corresponding increase in security often justifies the additional complexity in design.

An alternative approach would be to one-way map the password and time-variant parameter before transmission over the network. The host system one-way maps the plaintext password from secure memory and compares the result to the received value. If the two values are equal the user is authenticated, otherwise the authentication attempt fails. This alternative does not require the distribution of cryptographic keys, however it does require secure storage of plaintext passwords at host computers. Plaintext passwords could be encrypted under a secret storage key for additional protection.

4. TOKEN BASED AUTHENTICATION

4.1 Overview

The identity of a human user can be proven by requiring the user to demonstrate possession of a physical object which is unique to that user, or to a group of users. Objects used for this purpose are known as authentication tokens. For example, a driver's license would be considered an authentication token because it can be used to prove the identity of its owner. Tokens designed for use with automated authentication systems are encoded with information which is used in performing the authentication protocol required by the host system in order to verify the identity of the token's owner [5]. Since the uniqueness of the information stored on an authentication token is responsible for proving the identity of its bearer to the host system, the information must be protected against duplication or theft. Advanced tokens usually contain a microprocessor and semiconductor memory, and support sophisticated authentication protocols which provide a high level of security.

4.2 Form Factor

Authentication tokens are currently available in a variety of physical forms. The size, shape, and physical materials from which a token is manufactured are referred to collectively as the token's form factor. These parameters affect the durability, portability, security, and convenience for a given type of token. For example, some tokens have electrical contacts mounted on the outer surface of the token's casing. The electrical contacts are connected to an integrated circuit embedded in the token. When an electrostatic discharge of sufficient potential is applied to the contacts, the integrated circuit may be damaged. Care must be taken in the design of tokens with electrical contacts to minimize the risk of damage from static discharges, since the human body can accumulate a significant static charge in dry weather. To compensate for this, some types of tokens have contacts which are recessed in a conductive plastic casing [6]. This type of token is less susceptible to damage from stray static discharges, because the casing of the token absorbs the charge before it reaches the contacts. Other varieties of tokens have no electrical contacts, further reducing the risk of static damage. Each form factor involves trade-offs which must be evaluated for a specific application. Tokens with recessed contacts usually require a thicker casing than those with surface-mounted contacts, which can make the token more difficult to carry in a pocket. Customers can sometimes select from a number of different form factors with the same functionality, making it possible to choose the form factor which is best suited to a particular application.

4.3 Workstation Interface

Most authentication tokens require an electronic interface in order to communicate with the workstation during the authentication process. This interface is commonly known as a reader/writer, because it reads data from and writes data to the token. Reader/writers may be built directly into terminals or workstations, or they may be separate devices which are connected to a standard communications port or special purpose interface on the workstation [7,8]. Reader/writers which are built into workstations can provide a higher level of physical protection for the communications path between the workstation and the token, because there is no external cable which could be monitored by an attacker. However, this type of reader/writer is designed to work with the hardware of a specific host system and may not be compatible with other types of computers. If it is necessary to move reader/writers from one computer to another frequently, an external reader/writer which connects to a standard communications port will be more convenient.

Reader/writers vary in complexity and cost. Tokens which do not have a microprocessor are essentially data storage devices which contain the information required by a host system to verify the identity of the token's owner. Reader/writers designed for use with this type of token are usually microprocessor based, because the reader/writer must be able to perform a fairly complicated series of operations. In a typical implementation, the reader/writer reads authentication data from the token, and then uses this data to perform the authentication protocol required by the host system. Since these intelligent reader/writers have significant processing capabilities, they tend to be more expensive. However, the additional capabilities of intelligent reader/writers can be used to offload some of the processing burden from the host system. Some types of intelligent reader/writers can be programmed to work with a variety of host communications protocols, or to work with several different tokens.

Tokens which contain a microprocessor are often referred to as intelligent or smart tokens. Most smart tokens can perform communications functions such as data formatting, flow control, and error detection and recovery. Smart token reader/writers can be very simple, because the token typically requires only hardware-level support from the reader/writer in order to communicate with the host system. Since these reader/writers require fewer components than an intelligent reader/writer, they are often less expensive. Smart tokens can also work with microprocessor based reader/writers, in applications where the additional capabilities of an intelligent reader/writer are required. The use of intelligent reader/writers usually adds to the cost of the system, but this may be acceptable if the additional functionality provided by the reader/writer is needed.

For a specific application, the expected ratio of tokens to reader/writers is a major factor in determining the most effective overall cost distribution for equipment. In a situation where many low cost tokens will be used with a relatively small number of reader/writers, the higher cost of intelligent reader/writers is usually offset by the lower cost of the tokens.

Some tokens do not need a reader/writer, because the user acts as the communications link between the token and the host system. This type of token usually has an integral keypad and display for communications with the user. The user is required to manually transfer authentication data between the token and the user's terminal. Since these tokens operate without a physical connection to the terminal or workstation, they can be used in a variety of environments regardless of the type of terminal available. However, the user may have to repeat the manual authentication process each time the user logs on to a different host on the network, since host computers cannot communicate directly with the tokens. Tokens which use a reader/writer interface can automate the authentication process so that the user only needs to be involved in the initial authentication at the beginning of a session. Subsequent authentications can be performed automatically by the token as the user accesses different host machines.

4.3.1 Contact Interfaces

The types of interfaces between tokens and computers can be broadly classified as either contact or non-contact. The majority of tokens need to make actual physical contact with the reader/writer to perform data transfer. For example, magnetic stripe tokens (the kind used in automated teller machines) are inserted into a reader/writer so that the magnetic stripe makes contact with an electromagnetic sensing device. Most integrated circuit tokens require an interface in which electrical contacts located on the token physically touch matching contacts on the reader/writer in order to supply such functions as power, ground, and data signals. The physical arrangement and functional definition of these contacts has an impact on the interoperability of tokens and reader/writers, since these devices cannot communicate unless the contacts are defined in the same way.

No significant standards addressing the contact arrangement of authentication tokens existed until 1990. Token manufacturers relied on de facto standards, or developed their own proprietary specifications. This made it difficult in many cases to use tokens made by one vendor with reader/writers manufactured by another vendor. In 1990, the International Organization for Standardization (ISO) developed a standard for the dimensions and location of contacts on integrated circuit cards, known as ISO 7816-2 [9]. Integrated circuit cards are commonly known as smartcards. The ISO standard does not address contact specifications for other types of tokens. The majority of commercially available smartcards follow this standard, allowing for some degree of interoperability between the products of different manufacturers.

ISO 7816-2 specifies eight electrical contacts arranged in two parallel rows of four contacts each. The contacts are labelled C1 through C8, with the following assignments:

C1 - Supply voltage

C2 - Reset

C3 - Clock

C4 - Reserved

C5 - Ground

C6 - Programming voltage

C7 - Data input/output

C8 - Reserved

ISO places the contacts within 10.25mm of the left edge of the card, approximately centered between the top and bottom edges of the card. The supply voltage on C1 is typically +5 volts as referenced to ground (C5). C2 is a microprocessor hardware reset line. Most smartcards do not have an internal system clock, so an external clock signal is provided on C3. C4 and C8 are reserved for future use. Some smartcards contain Electrically Programmable Read-Only Memory (EPROM), or Electrically Erasable Programmable Read-Only Memory (EEPROM). Older EPROM and EEPROM technologies require a programming voltage several times greater than the power supply voltage. C6 provides the programming voltage for cards which need it during memory write cycles. Most of the smartcards available today use newer EPROM and EEPROM technology which operates on +5 volts. These cards do not require the programming voltage function of C6, because the electrical energy required to write data to memory is derived from the supply voltage on C1. Data exchange between the smartcard and reader/writer is accomplished through C7, which is a serial data connection. Data flows in both directions, and so protocols which avoid the collision of data on this line must be used. The ISO standard for contact dimensions and location provides a basis for interoperability between smartcards and reader/writers at the hardware level.

The token reader/writer provides the hardware level components of the communications path between the token and the host. However, the token and the host system must use a common format for data exchange. Smart tokens often use a serial data transmission protocol, although tokens are available which use parallel protocols. Smartcards usually follow ISO standard 7816-3 [10], which specifies electronic signals and transmission protocols. According to ISO 7816-3, a smartcard will respond to a hardware reset with a series of characters which specify, among other things, the format which will be used for data transmission. ISO refers to the transmission protocol type as T, which may be set to one of several values. T = 0 specifies an asynchronous half duplex protocol, which is used by the majority of smartcards at the present time. T = 1 is an asynchronous half duplex block transmission protocol which allows for more efficient transfer of large blocks of information. Although T = 1 is not used as often as T = 0 at present, the higher performance of T = 1 will be required as more powerful smartcards become available. ISO has reserved the protocol type T = 14 to indicate protocols which have not been standardized. Systems which conform to T = 14 are therefore not compliant with any of the specific transmission protocols defined in 7816-3.

4.3.2 Non-Contact Interfaces

Tokens are currently available which do not require physical contact with the electrical connections on the reader/writer. Non-contact interfaces increase the lifetime of tokens, and their convenience to the user. Most non-contact interfaces involve inductive coupling, capacitive coupling or a combination of the two methods to transfer electrical power and data. These interface technologies do not require physical contact with the circuitry of the token. However the token must still be inserted into the reader/writer since the two must be within relatively close proximity (a few centimeters). Some non-contact interfaces use optical coupling, whereby infra-red (IR) radiation is modulated to provide control and data signals to the token. IR tokens usually require a physical contact for the supply voltage, because IR links cannot transfer enough energy to generate the supply voltage for a token.

Another type of non-contact interface uses radio frequency (RF) signals to transfer information between the token and reader/writer. A token which uses an RF interface does not need to make physical contact with conductors in the reader/writer. Instead, the reader/writer combines power and data transmission signals into a low power electromagnetic carrier (EMC) wave which is received by a wire coil embedded in the token. Circuitry in the token derives both a supply voltage and data from the carrier. In most environments, it is not desirable to use high-power radio frequency carrier signals due to possible adverse health effects and electromagnetic interference with other systems. Since the total power available to the token from the RF signal is relatively low, RF tokens typically have less data storage and processing capability than tokens which use a contact interface. The majority of RF tokens which can operate at a significant distance from a reader/writer use a very simple authentication protocol. The reader/writer transmits an RF signal to the token, which responds by sending an identification number back to the reader/writer. This protocol is often used for identification of shipping containers and inventory control applications, where RF tokens can be attached to a physical object.

Authentication tokens which use an RF interface offer some advantages in terms of user convenience. When a user sits down at a workstation equipped with an RF reader/writer, the reader/writer can communicate with the user's RF token while it is still in the user's pocket. Since the user does not need to insert the token into the reader/writer, the authentication process is transparent to the user.

RF technology has some limitations which should be factored into the design of an authentication system which uses this type of token. Because authentication data is transmitted between the token and reader/writer via an RF signal, there is a significant risk that this data could be received and recorded by an unauthorized device monitoring the carrier frequency used by the token and reader/writer. A sophisticated attacker could construct a device which mimicks the functions of an RF reader/writer, or modify a reader/writer for this purpose. This device would be placed within the working range of a user's RF token, without the user's knowledge. The device would activate the user's token by transmitting an RF signal of the correct frequency, and capture the identification number returned by the token. Once a valid identification number is obtained in this manner, the attacker could fabricate an RF device which would allow the attacker to pose as a legitimate user.

There are several methods which can provide protection against this type of attack. Users can be required to enter a password which is independent of the identification number stored on the user's RF token. In this case, user authentication would depend on knowledge of a password and possession of a valid token. An attacker would then be faced with the difficult task of guessing the user's password, in addition to fabricating or stealing the user's RF token. Another approach is to use an RF token with enough processing power to implement secure authentication protocols in which the data exchanged between the token and reader/writer is different for each authentication. RF tokens with significant processing power are difficult to design and manufacture using current methods, but advances in technology should make these products more available in the near future.

4.4 Processing Capability

The processing capability of various authentication tokens ranges from simple data storage to implementation of sophisticated cryptographic authentication protocols. All tokens must be capable of storing the information used to authenticate to a host system in some form. In applications which require minimal security, authentication data can be stored as a physical pattern, such as a series of holes punched in a plastic card. When this card is presented to an access control system, the system examines the pattern of holes and compares it to a list of currently valid patterns. If the pattern is determined to be valid, the user is granted access to whatever resource the access control system is protecting. However, it may be relatively easy for an attacker to duplicate the physical pattern and thus create a counterfeit authentication token. Because of this threat, systems which rely on a physical pattern for user authentication often require the generation of new authentication patterns and issuance of new authentication tokens at frequent intervals to reduce the amount of time an attacker has to produce a counterfeit token. Authentication tokens with a higher level of processing power can provide greater security in many applications, but the relative cost of a token tends to increase with processing capability.

4.4.1 Memory Tokens

If the authentication data is stored in a magnetic, electronic, or optical form, more sophisticated methods are required to decode the data. This makes it more difficult for an attacker to duplicate the token, because the attacker must understand the interface between the token and the host system in order to extract the authentication data and store it on a counterfeit token. Magnetic stripe tokens fall into this class, as do tokens containing integrated circuit memories. The data stored on these tokens is often encrypted to provide additional protection from disclosure. The encryption process is typically based on the user's password or Personal Identification Number (PIN), so the authentication data cannot be decrypted unless the user's password or PIN is known.

Memory tokens based on semiconductor technology are essentially memory chips mounted in a package which is more durable than the standard Dual Inline Package (DIP) used for most integrated circuits [11]. In addition, memory tokens use contact or non-contact interfaces designed to operate with token reader/writers. The reader/writer provides access to the control and data lines of the integrated circuit, so that a host system can read information from and write information to the token. The primary difference between a semiconductor memory token and a standard computer memory chip is that a user can easily remove the token from a host system. The user can therefore exercise some degree of physical control over the information stored on the token.

Most memory tokens use EPROM or EEPROM as the primary storage medium. These memory technologies are widely used in electronic products, including computer systems. Both EPROM and EEPROM are in-circuit programmable nonvolatile memory technologies, meaning that data which is written to memory locations will be retained when system power is turned off. Most microprocessor based tokens contain a small amount of Random Access Memory (RAM) for use as a scratchpad area. However, some memory tokens also use RAM technology for long term storage of data across multiple sessions. Since RAM retains data only as long as power is applied, tokens which use RAM for nonvolatile storage typically contain a battery to avoid loss of data when external power is withdrawn.

The type of memory technology used in a token is an important factor in the design of token based authentication systems. EPROM is a write-once technology, where data can only be written to a specific memory location once. When all memory locations have been used, no new data can be stored on the chip. However, most EPROM chips can be erased by exposure to ultraviolet light and re-used. For security reasons, tokens which use UV-eraseable semiconductor memory often have features which prevent the erasure of data in this manner. EPROM tokens can eventually run out of storage space when data has been written to all memory locations. These tokens are most useful in applications where the data stored in the token's memory does not need to be updated frequently.

EEPROM technology combines the data retention characteristics of EPROM with the ability to re-use memory locations. Current EEPROM technologies are rated at ten thousand write cycles per memory location, and will retain stored data for ten years. These ratings are more than adequate for most applications involving authentication tokens, since the expected lifetime of a token is usually much less than ten years. EEPROM tokens are useful in applications where data must be modified often, because the token will not become unuseable when all memory locations have been filled. Many applications require the ability to change data stored on authentication tokens, such as user passwords or cryptographic keys. Some tokens store audit trail data which is updated frequently.

Memory tokens based on optical storage technology can store large amounts of data, often in the range of several megabytes. Data is read from optical storage tokens with circuitry similar to that used in commercial audio compact disc players. However, writing data to this type of token requires comparatively sophisticated and correspondingly expensive equipment. Optical tokens are most often used in applications which require high storage capacity and infrequent updates to information stored on the card. The amount of data which a token must store and the number of times this data will be modified are critical factors in determining the type of memory technology best suited to a particular application [11].

4.4.2 Microprocessor Tokens

The most sophisticated integrated circuit tokens contain a microprocessor in addition to semiconductor memory. Microprocessor based tokens are often referred to as smart tokens, since they have some degree of data processing capability. Smart tokens have some unique features which can enhance the security of an authentication system. The microprocessor of a smart token can control access to sensitive data stored on the token. Many smart tokens require the submission of a password or PIN, or some other form of authentication, before the token will allow a host system to read data from or write data to the token's memory. The microprocessor acts as a gateway between sensitive data stored on the token and the host system, providing a higher level of protection for the data than can be attained with memory-only tokens. Some smart tokens have hardwired control logic designed to perform a small number of relatively simple functions, such as password checking and data transfer. However, the majority of smart tokens contain a more general purpose microprocessor which can execute programs stored in the token's memory. Executable programs stored in nonvolatile memory are referred to as firmware. Smart tokens with sufficient processing power and storage space can implement cryptographic algorithms in firmware. A smart token with cryptographic capabilities is a very effective tool for implementing secure authentication protocols.

Before smart tokens with EPROM or EEPROM became available, firmware was usually stored in Read-Only Memory (ROM). The contents of ROM are fixed during the semiconductor manufacturing process, and cannot be changed thereafter. This characteristic of ROM provides protection against inadvertent or malicious modification of data and executable code. However, a system designer may wish to modify smart token firmware during the development phase of a project. If the smart token architecture requires that firmware be stored in ROM, the designer must use a software simulator or hardware emulator for firmware development.

Software simulators are effective in some situations, but they often do not provide a realistic representation of the behavior of the smart token hardware in real time. Hardware emulators can accurately portray the dynamics of smart token hardware, but these devices tend to be expensive and somewhat complicated to use. Some smart tokens can store firmware in EPROM or EEPROM, making it possible to change the firmware during the development process. Tokens which can be reprogrammed in this manner are useful in situations where system specifications are subject to frequent change during the development and prototyping cycles, since the token firmware can be modified to meet different specifications. Precautions should be taken to insure that smart token firmware can only be modified by authorized parties, since the firmware plays a critical role in the security of an authentication system. Microprocessor tokens typically also contain a small amount of RAM for use as working storage during a session.

Smart tokens which contain EPROM or EEPROM can be vulnerable to attacks which take advantage of the electrical characteristics of these memory technologies. Each time data is written to memory, a pulse of electrical energy must be supplied to the token through the programming voltage contact or the supply voltage contact. By monitoring these pulses, it is possible to detect the start of memory write cycles. If power to the token is withdrawn as soon as the beginning of a write cycle is detected, the token may not be able to complete the write cycle.

In some cases, an attacker may be able to use this process to circumvent the security of the token. Smartcards, for example, often require the submission of a password or PIN to verify the identity of the cardholder before the card will perform any subsequent operations. If an attacker manages to steal one of these cards from a valid user, the attacker will not be able to use the card to gain access to a host system because the card will not operate unless it receives the user's password. In this situation, the attacker will probably attempt to guess the user's password through exhaustive search techniques.

Many smartcards have a protection mechanism which will render the card inoperable if some number of incorrect passwords are submitted in sequence. This number is usually small enough that the chances of guessing the password before the card becomes inoperable are insignificant. However, the smartcard must store a count of the current number of consecutive failed password submissions. This count must be incremented each time another consecutive failure occurs, and is therefore normally stored in EPROM or EEPROM. In this situation, an attacker may be able to take advantage of the information provided by the electrical pulse of energy which occurs during memory writes. The attacker can submit passwords to the card, and immediately withdraw the supply voltage each time the card attempts to increment the counter which keeps track of password failures. Since the card will not be able to update the counter before power is withdrawn, the mechanism for limiting the number of consecutive password failures will not operate. Given enough time, the attacker could complete an exhaustive search to find the correct password. Submission of the correct password would not cause the card to update the counter, and the attacker would be able to detect this since the electrical pulse of energy associated with a memory write would not occur.

The attack described above is not limited to password guessing. Smart tokens often need to store information in nonvolatile memory which indicates the successful completion or failure of a security-critical process. For example, most tokens perform a series of exchanges with a host computer during the user authentication process. The token must keep track of which authentication steps have been completed successfully so that subsequent steps cannot be executed in the wrong sequence. If an attacker can keep the token from accurately recording sequence information, the security of the system may be compromised. However, a relatively simple mechanism can be implemented which reduces the risk associated with this type of attack. If the token writes data to memory regardless of the success or failure of an operation, an attacker will not be able to distinguish between a positive or negative result. In the case of password guessing, the attacker might still be able to keep the token from updating a counter representing the submission of failed passwords. However, the attacker would never know when the correct password was submitted because the token would attempt a memory write in all cases. Many smart token manufacturers use this protection mechanism in their products.

4.4.2.1 Hand Held Password Generators

One time password generators and handheld challenge response calculators are microprocessor based authentication tokens which do not require a physical connection to host systems. These devices communicate directly with human users through an onboard display and some form of keypad. Users relay authentication data, such as passwords or encrypted challenges, between tokens and host systems manually. The following discussion explains the operation of these devices in general terms, but many variations are possible.

One time password generators create a sequence of passwords which are synchronized in some manner with host systems. Each password is only valid for one authentication, and so cannot be recorded and replayed to gain access. Synchronization is often based on a secret initial seed value which is permuted at specific time intervals, or possibly each time an authentication event occurs. Without knowledge of the secret value and the number of times it has been permuted, an oberver cannot predict the next password in the sequence even if one or more previous passwords are known. Some password generators require the user to enter a PIN via the onboard keypad before the device will generate a password.

Challenge response calculators accept a random challenge from the host system, which is read by the user and entered through the calculator's keypad. The calculator then encrypts the challenge with a secret cryptographic key and displays the result. The user enters the encrypted challenge on the host keyboard, and the host verifies the encryption. This process requires each participant in the authentication process to possess a copy of the secret cryptographic key. Challenge response calculators typically require users to enter a PIN before executing the authentication exchange with a host system.

Authentication tokens that do not require an electronic interface for communications with host systems can be used in a wide range of environments, since all general purpose computer terminals can display random challenges and accept passwords or encrypted challenges as keyboard input. One time password generators and challenge response calculators eliminate the cost of the special purpose interfaces required by other types of tokens.

4.4.3 Multi-Application Tokens

Some authentication tokens are capable of serving more than one application. Tokens with sufficient memory capacity and processing power can implement several authentication protocols, to accommodate host computer systems which use these different protocols. Multi-application tokens can also support functions which do not relate directly to the security mechanisms of a system, but increase the level of convenience for system users. For example, a user may wish to store a configuration file on an authentication token which is read by a host system during the login process to customize the user's environment. Another possible application involves the process of tracking personnel records and medical information, which can be particularly difficult in an environment where people are transferred frequently or work in different locations on a daily basis. Medical records or other personal data can be stored on tokens, making it easy for users to carry job related information from place to place. Authentication tokens have the additional benefit of built-in mechanisms to protect information stored on the token from unauthorized access.

Memory tokens can support multiple applications in a straightforward manner. Each application treats the tokens as data storage devices containing information which is specific to the owner of the token and the particular application. However, there are some drawbacks to this approach. Since memory tokens have no processing power, they cannot enforce separation of data used by different applications. Since all the available data can be accessed, one application could modify the data owned by another application. The applications must therefore be trusted not to access data on the token owned by another application unless the access is explicitly allowed. Particularly when applications are running on different host systems, this creates problems in terms of defining which application can access which sections of the token memory, and in enforcing this definition. Memory tokens can be used in a multi-application environment if applications running on the host system incorporate the proper controls to insure that data stored on the tokens is accessed in the proper manner.

Microprocessor tokens can be programmed to deal effectively with the requirements of a multi-application environment. Since the firmware of a microprocessor token controls access to data stored in the token's memory, the token itself can determine which applications are allowed to access specific data storage areas on the token. System designers can shift the responsibility for managing the low level details of secure data storage to the token firmware, so that host applications can use the services provided by the token in a standard manner. It is possible to implement a hierarchical file structure in the memory of a token which consists of multiple directories, each owned by a different application. Since this type of file structure is common in many computing environments, applications can access data stored on a token in much the same way that files stored on other types of nonvolatile media are accessed. Depending upon the host environment, the only custom software required might be a device driver designed to communicate with the token reader/writer, since these devices normally use a communications protocol which is different from the protocols used by standard system peripherals. An intelligent reader/writer could be designed to emulate the functions of a standard peripheral, such as a disk drive. This would eliminate the need for a custom device driver, since the system would be able to communicate with the reader/writer through a standard device driver.

Token based authentication provides a relatively high level of security with minimal inconvenience to system users if implemented correctly. However, users sometimes perceive authentication technology as more of an impediment than an essential requirement for the protection of valuable resources. The value-added aspect of multi-application tokens can encourage the use of secure token based authentication systems, because these tokens have the ability to perform a variety of tasks which contribute to user acceptance. The primary factors which limit the number and complexity of the applications which a token can support are the total memory capacity and processing power of the token. In some cases a multi-application token may not be as well suited to the needs of a specific application as a single purpose token.Multi-application tokens can also be more expensive due to the additional capability. As token technology progresses, these factors will become less of a limitation.

4.5 Recommendations

From a security standpoint, the strength of token based authentication lies in the fact that the device containing the information which verifies the identity of the user is portable. Therefore, the authentication information can be kept in the user's possession. The greatest threat to the security of this type of system is the possibility that an attacker could steal a valid token in order to pose as an authorized user. A sophisticated attacker might also be able to counterfeit a token. These threats can be reduced by requiring the user to submit a password or PIN when the token is used for authentication. There should be a limit to the number of consecutive incorrect PIN submissions, in accordance with section 3.2.7. Without the password, a stolen or counterfeited token would not allow an attacker to gain access to the system. This can significantly increase the level of assurance since a user's identity is verified based on something the user knows in addition to possession of the token. Compromise of one user's token should only compromise that user, not the entire system. Use of tokens with cryptographic capabilities can contribute greatly to the security of an authentication system. In addition, the use of passwords or PINs in combination with authentication tokens is recommended for applications where this approach is practical.

The type of authentication token selected for a particular application will depend on a number of factors. The simpler tokens which store data on a magnetic stripe tend to be less expensive, but may require relatively complex interface devices. These tokens are often easy to counterfeit, because authentication data can be read from the token by anyone with an appropriate interface. Integrated circuit tokens are generally more expensive, particularly when they contain microprocessor circuitry. However, microprocessor tokens can provide a very high level of security at a reasonable cost.

4.6 Example - The NIST Advanced Smartcard Access Control System

NIST has developed an Advanced Smartcard Access Control System (ASACS) in collaboration with several commercial vendors. This system is described in "An Overview of the Advanced Smartcard Access Control System (ASACS)"[12], and several related documents [13,14,15,16,17,18,19]. A condensed version of the ASACS system overview is presented here, as an example of one approach to the development of a smartcard based authentication system. The primary goal of the ASACS project was to develop an advanced smartcard system which exploits recent advances in semiconductor and cryptographic technologies for secure login authentication. ASACS also provides secure data storage, automated key management, and digital signature capabilities. The services supported by the ASACS implementation are designed for use within networking environments, including both local area networks and wide area networks such as the Internet.

The ASACS smartcard provides cryptographic capabilities based on standard cryptographic algorithms and techniques, in combination with software running on a host computer. Many of the underlying concepts applied to the design of ASACS have been successfully demonstrated in the NIST/Datakey Token Based Access Control System (TBACS) [6] as well as the Smartcard Access Control System (SACS) [20] projects. Each of these systems provides token-based secure access to a host computer through a cryptographic handshake protocol based on the Data Encryption Standard (DES) algorithm. However, the ASACS project involves the development of a smartcard with greater capabilities through the addition of public key cryptographic functions. A new smartcard reader/writer with significantly greater capabilities has also been developed for ASACS. The ASACS reader/writer has computational capabilities, and includes a microprocessor, programmable memory, a keypad, and an LCD display. These features support the needs of mobile users who require a portable reader/writer for authentication from remote sites. To demonstrate the capabilities of ASACS, several applications have been developed, most notably a system maintenance program and several other useful demonstration programs. In addition, ASACS has been integrated with the Privacy Enhanced Mail (PEM) system.

4.6.1 System Overview

Figure 1 depicts the ASACS system components. A user possessing a smartcard inserts the card into the reader/writer which is attached to a local workstation. The workstation is connected to a local area network (LAN), which in turn may be connected to other networks. The smartcard may be used to control the user's access to both the local workstation as well as to other workstations and host computers on the attached networks.

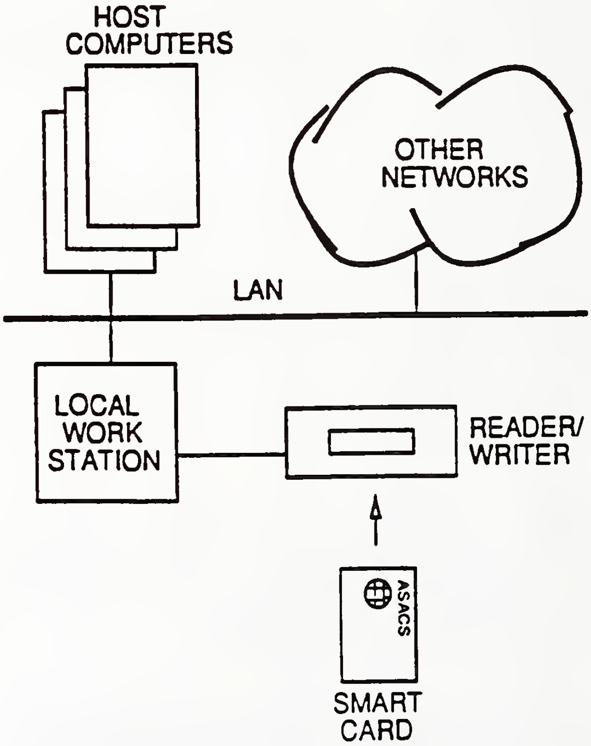


Figure 1: ASACS system components

From an architectural standpoint, ASACS is divided into several different functional layers, comprising both the hardware and software components of the system (Figure 2). The lowest layer consists of the ASACS hardware, including the public key smartcard and either the SACS reader/writer or the ASACS portable reader/writer. The next layer of ASACS is comprised of host system software, which is functionally divided into four layers. This software provides a convenient and standard method for integrating the ASACS public key smartcard into a wide variety of host system application software. The top layer is a Smartcard Application Program Interface (SCAPI) which is directly accessed by applications software to interface with the ASACS system. The other layers provide command set interfaces for the smartcard commands and the reader/writer commands, a smartcard communications protocol, and hardware-level I/O support.

Finally, the top layer of ASACS represents the various applications with which the ASACS system can be integrated. ASACS can be integrated into these applications using either the SCAPI or the command set interfaces. A security officer maintenance program and several demonstration programs, including a signature utility program and a login manager, were developed as a part of the ASACS project. In addition, using the SCAPI, the ASACS system has been integrated into an implementation of the Internet Privacy Enhanced Mail system.

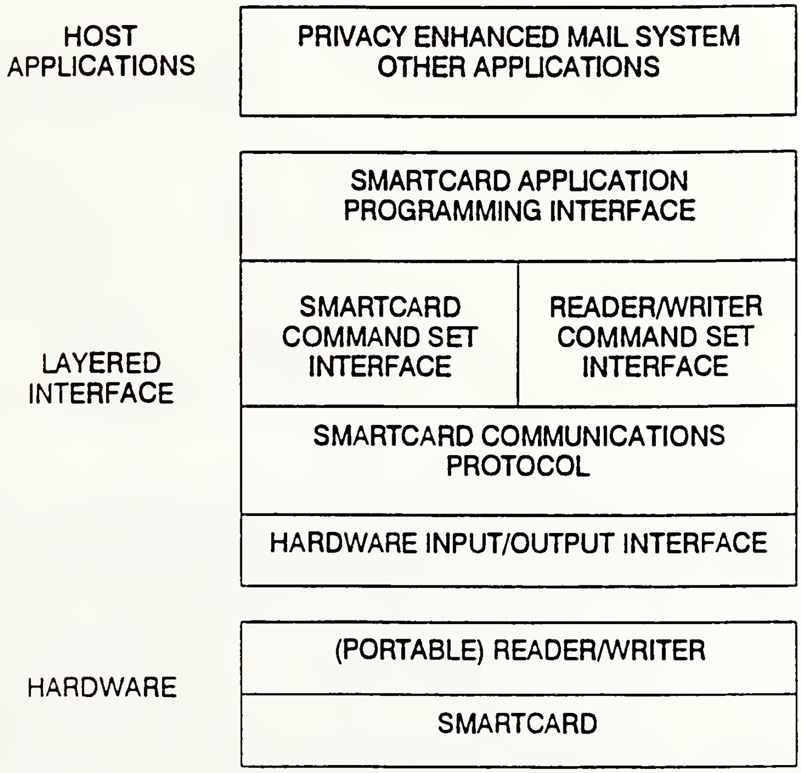

HOST APPLICATIONS
PRIVACY ENHANCED MAIL SYSTEM
OTHER APPLICATIONS
SMARTCARD APPLICATION PROGRAMMING INTERFACE
LAYERED INTERFACE
SMARTCARD COMMAND SET INTERFACE
READER/WRITER COMMAND SET INTERFACE
SMARTCARD COMMUNICATIONS PROTOCOL
HARDWARE INPUT/OUTPUT INTERFACE
HARDWARE
(PORTABLE) READER/WRITER
SMARTCARD

Figure 2: ASACS layered system architecture

4.6.2 The ASACS Smartcard

The ASACS smartcard is based on the Smartcard Access Control System (SACS) developed by NIST under a previous contract. The SACS and ASACS smart cards contain an integrated circuit microprocessor designed specifically for smart card applications [21]. This processor is configured with 256 bytes of RAM, 10K bytes of ROM, and 8K bytes of EEPROM. In order to meet ISO requirements for contact spacing and arrangement, the processor has pads for power (+5V), ground, clock (10MHz), reset, and serial I/O [9]. An ISO-standard micromodule is bonded to the processor, and this assembly is then mounted in a plastic card with the same dimensions as a standard credit card.

4.6.2.1 Smartcard Firmware

The ASACS public key smartcard firmware implements a set of commands which support card maintenance, key management, user authentication, data storage, and data encryption and authentication. Access control software running on a host computer issues commands to the smartcard through the reader/writer interface. The firmware of the card then executes the requested function and returns the appropriate response to the host computer. It is the responsibility of the host access control software to mediate the authentications between the user, the user's smartcard, and the host computer.

The ASACS command set is the successor to the smartcard command set developed for the Smartcard based Access Control System (SACS). The cost and time constraints of the ASACS project did not allow for the production of a new ROM mask. Therefore, the ROM mask developed for the SACS project was also used for the ASACS smartcard. ASACS retains the symmetric key capabilities of the original SACS system, since the authentication protocol is based on the DES algorithm. This challenge-response authentication protocol provides a rapid and secure method for two parties to perform mutual identity verification based upon the possession of a shared secret key and the use of that key to encrypt randomly generated cryptographic challenges. This protocol is described in detail in NIST Special Publication 500-157 [22]. The ASACS smartcard is capable of accepting or generating the initial cryptographic challenge, and therefore complies with the requirements of ANSI X9.26 [23] for secure sign on.

The principal difference between the ASACS and SACS command sets is the addition of public key cryptographic capabilities. There are certain arithmetic operations, such as modular exponentiation and modular multiplication, which are common to a variety of public key algorithms. These operations have been implemented in the ASACS firmware as distinct routines which can be used to support most of the currently available public key algorithms. The development and optimization of firmware which performs these modular operations is the most difficult aspect of implementing public key cryptography on a smartcard. A variety of public key algorithms can be realized in the ASACS smartcard firmware by calling the low-level arithmetic routines in the required sequence. Both the Digital Signature Algorithm (DSA), which has been proposed by NIST as a Digital Signature Standard (DSS) [24], and the Rivest-Shamir-Adleman (RSA) [25] cryptographic algorithm have been implemented in the

ASACS smartcard firmware.

Figure 3 depicts the layout of the ASACS smartcard memory from a high level perspective. The majority of the firmware is stored in ROM, including a bootstrap routine and code for the commands from the SACS smartcard. The DES [26] algorithm is also located in ROM. The EEPROM contains the firmware for the public key algorithms, a command interpreter, and a jump table which points to the firmware routines associated with each command. Since the addresses in the jump table can be modified, new firmware routines can be loaded into EEPROM to replace existing routines and to add new functions. Specific locations in EEPROM are reserved for the storage of symmetric and asymmetric key components. In addition, a number of general purpose data storage zones are available in EEPROM. See [13] for a more detailed description of the ASACS public key smartcard.

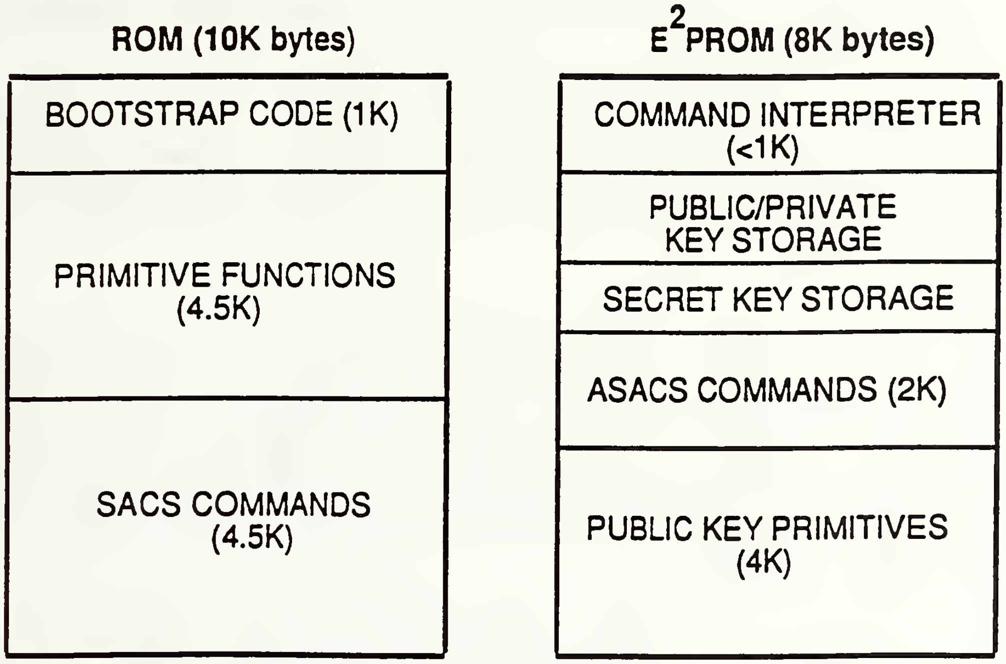


Figure 3: ASACS smartcard memory layout

4.6.3 Smartcard Reader/Writer

The ASACS public key smartcard can be interfaced to a workstation using either the SACS reader/writer or the new ASACS portable reader/writer. Both the SACS and the ASACS reader/writers provide an RS-232 serial communications connection between the smartcard and the host computer. RS-232 was chosen because a serial port is standard equipment on the majority of computers. Therefore, the reader/writer can be connected to most computers without the need for a custom interface or hardware modifications.

4.6.3.1 SACS Reader/Writer

The SACS reader/writer is a relatively unsophisticated device which simply serves as a direct I/O interface between the smartcard and a host. It cannot perform any processing itself since it does not contain a microprocessor. Its main purpose is to provide power, ground, clock and I/O signals to a SACS or an ASACS smartcard. To interface the smartcard to the host, the reader/writer performs level conversion between the 12V RS-232 I/O signals used by the host and the 5V I/O signals used by the card. See [7] for a more detailed description of the SACS reader/writer.

The SACS reader/writer features an ISO standard smartcard receptacle, external power and data indicator lights, and an RS-232 port for connecting to a host. In addition, the SACS reader/writer's card receptacle features a locking mechanism which holds the card internally after insertion into the reader/writer, and an automatic ejection mechanism to remove the card from the reader/writer.

An RS-232 cable is required to attach the SACS reader/writer to a host, whereupon it functions as data communications equipment (DCE).Signals are sent by the reader/writer to the host which indicate that the reader/writer is powered-up and that a card is inserted. The SACS reader/writer is a rectangular box approximately 2 1/2 inches high, 5 inches deep, and 5 inches wide. An ISO smartcard receptacle and indicator lights are located on the front of the reader/writer, and the power cord and RS-232 jacks in the rear. The power supply for the SACS reader/writer is internal.

The SACS reader/writer is designed to accept a smartcard whose physical characteristics, dimensions and contact locations adhere to ISO 7816, Parts 1 and 2 [9,27]. The electrical signals that the SACS reader/writer supplies to the smartcard also meet most of the requirements specified in ISO International Standard 7816, Part 3 [10], with the exception of the initial clock (CLK) frequency, which is 10MHz as opposed to 3.5795.

4.6.3.2 ASACS Portable Reader/Writer

The ASACS portable reader writer was built to provide functionality not offered by the earlier SACS reader/writer. As a portable device, it allows users the option to authenticate themselves using hosts not equipped with a smartcard reader/writer. Several significant improvements have been made to the design of the reader/writer. The overall size has been reduced to less than half that of the SACS reader/writer, so that the device can easily be carried for use at remote sites. The new reader/writer is powered by rechargeable batteries, and includes a transformer for use with 110V line power. The front panel has a keypad and liquid crystal display which allow the user to interact directly with the smartcard. This feature is useful in situations where the reader/writer cannot be connected to the user's workstation. A protocol has been developed which allows the user to perform authentications manually via the keypad and display. A remote host computer can then require manual ASACS authentication even if the user's workstation is a terminal with no processing capability. In this case, all interactions with the card are through the keypad and display. After the user personal identification number (PIN) has been submitted to the card, the remote host will generate a random challenge and send this to the user's workstation. The user reads this challenge from the screen and types it on the reader/writer keypad. The smartcard encrypts the challenge and displays the encrypted result, so that the user can submit it to the remote host. When a serial connection to the workstation is available, the user still has the option of entering the PIN through the keypad on the reader/ writer. Since the user's PIN does not travel through the workstation, system security is enhanced.

The ASACS reader/writer has an 8-bit microprocessor with 256 bytes of internal RAM. In addition, the reader/writer has 256 bytes of EEPROM used for data and setup parameter storage, 32K bytes of RAM used for scratch pad and data buffering, and an industry standard 32K byte EPROM chip which holds firmware implementing the internal logic and external commands. The EPROM chip can be easily removed for custom firmware development. See [8] for detailed specifications for the ASACS portable reader/writer and firmware.

The reader/writer supports a set of commands that are executed directly on the reader/writer, as opposed to on the smartcard. These commands use the same protocol that is used for smartcard commands. Several of the reader/writer commands allow the host to load the default parameters into the reader/writer's non-volatile memory to control such things as baud rate, and the date/time. These same default values can also be specified manually from the keypad by pressing the F1 key to access the reader/writer's set-up menu. Another command can be used by the host to determine if a smartcard is inserted into the reader/writer. Two commands can be used to temporarily put the reader/writer in manual keypad entry mode. The first of these two commands, as discussed above, is used by the host to allow the user to enter their PIN to the smartcard via the reader/writer's keypad. The latter command can be called to allow the user to perform a manual challenge/response with a remote host. The remaining reader/writer commands can be used by the host to utilize the ASACS reader/writer's communications buffer for more efficient DES encryption, DES decryption or MAC calculation with the smartcard.

4.6.4 Smartcard Layered Interface

The ASACS host system software is comprised of a set of four interface layers. Each layer corresponds to a specific set of functions needed to integrate the ASACS system into a software application on a host system (see Figure 2).

4.6.4.1 Smartcard Applications Program Interface

The Smartcard Application Program Interface (SCAPI) [14] was developed to provide a consistent, but robust interface designed to ease the integration of smartcard technology into applications. The SCAPI is intended to insulate applications from the differences among the various smartcards, as well as differences likely to appear as smartcard technology evolves. The SCAPI is not tied to specific smartcards or to specific capabilities (e.g., memory capacity) of smartcards. In fact, the SCAPI can be, and has been, completely implemented in software, thus providing a simple, but useful tool for integrating smartcard technology into applications. The functional capabilities of a particular smartcard determines how much of the SCAPI functionality is implemented in software on the host computer and how much is performed on the smartcard. Thus, as technology advances, more of the SCAPI functionality may be directly implemented on the card or on the reader/writer while leaving applications unaffected.

The SCAPI currently defines four types of functions:

Initialization Functions,
Account Functions,
Cryptographic Functions, and
File and Directory Functions.

The SCAPI is intended to be consistent with the ANSI C standard. The file functions are designed to map directly onto those defined by Kernighan and Ritchie [28]. Since C is known for its portability, it makes sense to extend this platform independence to smartcard systems. Further, this flexibility and consistent feel for C programmers is likely to promote the use of the SCAPI. The directory functions reflect widely used operating system calls. Unfortunately, ANSI C does not address the cryptographic functionality to which smartcard technology is so well-suited. Therefore, the SCAPI defines a set of cryptographic functions which provide an algorithm-independent interface for cryptographic operations which may be implemented on a smartcard.

4.6.4.2 Smartcard and Reader/Writer Command Set Interfaces

The Command Set Interface Layer consists of C language object module libraries. The libraries each provide a set of C function calls, each directly corresponding to a command from the firmware command sets for the public key smartcard [15] and the portable reader/writer [16]. The function which represents a particular command is called with the appropriate input data for that command as arguments. The function returns the output data from the command and a status code. Status codes are mapped onto a set of error messages defined in a header file. This layer is called indirectly through the SCAPI, thus making the choice of reader/writer invisible to the application.

4.6.4.3 Communications Protocol and Hardware I/O Interface

The Smartcard Communications Protocol Layer transmits the data assembled by the Command Set Interface Layer to the ASACS portable reader/writer and the public key smartcard. The data is transmitted according to the communications protocol used by both the reader/writer and the smartcard. The Communications Protocol Layer interacts with the Hardware I/O Interface in order to send and receive each byte of the data.

The Hardware I/O layer consists of a software driver which provides low-level input/output routines for communicating with the smartcards. Currently, the Hardware I/O Layer consists of a serial interface, since both the SACS and ASACS reader/writers employ serial interfaces. This layer can support other types of hardware interfaces for reader/writers that do not employ an RS-232 interface.

The Serial I/O Interface is written to be as portable as possible across a broad range of hardware/software platforms, such as SUNOS (Sun's UNIX Operating System) and MSDOS. However, some systems may require that this layer be customized. The interface to this layer is clearly defined, and can be modified with minimal effort.

4.6.5 Applications Software

4.6.5.1 Security Officer Maintenance Program

The Security Officer Maintenance (SOMAINT) Program [17] provides functions which are used by a security officer or system manager. These functions include the initialization of cards for new users, synchronization and maintenance of key databases stored on the cards and host computers, deactivation of cards, and reactivation of cards which have been inadvertently deactivated or corrupted. The programs which support the system management functions are restricted to use by authorized security managers through the standard UNIX operating system file protections.

4.6.5.2 Signature Utility Program

The DSS Signature Utility Program [18] was developed to demonstrate the generation and verification of digital signatures using the ASACS public key smartcard. The program utilizes the hash algorithm specified in the Standard Hash Standard (SHS) [29] to calculate a hash value on a file of arbitrary size. The hash value is transmitted by the host computer to the smartcard, which applies the Digital Signature Algorithm (DSA) to this value to generate a digital signature with the cardholder's private key. The signature can then be verified by the host computer or the smartcard using the cardholder's public key.

4.6.5.3 Login Manager

The ASACS Login Manager [19] is a collection of programs which control login access to host computers. These programs manage the series of authentications between the user, the smartcard, and a host computer. When a user requests access to the host, the login manager establishes communications with the user's card through the reader/writer. The login manager prompts the user for the user PIN, and transmits it to the card in order to authenticate the user to the card. The card and host will then authenticate to each other using a random challenge-response protocol based on the DES. This protocol provides a means for rapid authentication of two parties with protection from wiretapping and playback attacks. If the authentications are successful, the user is granted a session on the host.

The login demonstration software also supports login authentication to remote host computers. When a system user wishes to access a remote computer, the user executes a program which communicates with the user's card to obtain a list of host computers with which the card shares authentication keys. This list of host computer names is displayed in a menu, so that the user can select the particular host to access.The software establishes a connection with the ASACS authentication server process running on the remote host selected by .the user. The remote host then performs the challenge-response authentication with the user's card in order to verify the identity of the user.

4.6.5.4 Privacy Enhanced Mail

The Internet Privacy Enhanced Mail (PEM) protocols are an extension to the existing Internet electronic mail protocol (RFC 822) which provide simple end- to-end security services including optional message confidentiality, message integrity, and source authentication with non-repudiation. The protocols are described in a 4 part series of Internet Requests for Comments [30,31,32,33].

The PEM security services are provided through the use of standard cryptographic techniques, including message encryption using the DES in the Cipher Block Chaining (CBC) mode of operation to protect message text and the RSA algorithm to provide for distribution of DES keys, digital signatures using RSA algorithm in conjunction with either Message Authentication Code (MAC), Message Digest Algorithm MD2 [34], or the Message Digest Algorithm MD5 [35]. RSA public keys are managed as public key certificates using a distributed certification hierarchy based on CCITT X.509 [36].

The TIS Privacy Enhanced Mail (TIS/PEM) System is a UNIX-based implementation of PEM [37]. At the core of the TIS/PEM system is the Local Key Manager (LKM), which, as its name implies, is responsible for all the local key management activities on a multi-user host system. This includes (1) maintaining a database for local users' private keys, (2) controlling the use of private keys to compute digital signatures and decrypt message tokens (encrypted message encryption keys), (3) maintaining a database for local and remote users' public key certificates, and (4) providing access to validated public key certificates. In addition, the LKM shares the responsibility for the registration of a local user, that is, the generation of a public/private key pair and the construction and digital signing of a certificate embodying the public key.

The ASACS system was integrated with the TIS/PEM system by integrating it with the LKM. In particular, a user's private key is generated by the LKM and then stored on the smartcard, where it remains in the protected confines of the smartcard. When called upon to perform the cryptographic operations involving the user's private key, the LKM, instead of performing those operations directly, now invokes the functions of the smartcard via the SCAPI. The smartcard then performs the necessary computation of a digital signature or decryption of a message token, using the private key stored on the smartcard.

The storage of a user's private key provides added protection that cannot be achieved in a shared database. The inherent security features of the smart card restrict access to the private key to the user, who must authenticate to the card before the private key can be used.

5.0 BIOMETRIC BASED AUTHENTICATION

5.1 Overview

Certain physical features of the human body are relatively unique from individual to individual. Facial photographs and fingerprints have long been used for personal identification, particularly by law enforcement agencies. Biometric authentication is the measurement of a unique biological feature used to verify the claimed identity of an individual through automated means. The biometric authentication mechanism will strive to measure a unique biological feature to the degree that only one person may be authenticated as a specific user. The biological feature may be based on a physiological or behavioral characteristic. The physiological characteristics measure a physical feature such as a fingerprint or face. The behavioral characteristics measure your reaction or response such as your signature or voice. The most used biometrics are fingerprint, retinal and voice authentication devices.

5.2 How Biometric Authentication Systems Function

The biometric authentication mechanism typically has two modes: enrolling and verifying. For initial use of the biometric, each user must be enrolled by a system administrator who verifies that each individual being enrolled is an authorized user. The enrolling process is the storing of an individual's biological feature (physical characteristic or personal trait) to be used later to verify the user's identity.

The biological feature is typically acquired by a hardware device which is at the front end of the biometric authentication mechanism. The front end component for these systems is a device known as a sensor. When a physical feature is presented to the sensor, the sensor produces a signal which is modulated in response to variations in the physical quantity being measured. If, for example, the sensor is a microphone used to capture a voice pattern, the microphone will produce a signal whose amplitude (voltage or current) varies with time in response to the varying frequencies in a spoken phrase.

Because the signals produced by most biometric sensors are analog in nature, it is necessary to convert these signals into a digital form so that they can be processed by computer. An analog to digital converter is therefore the next stage in most biometric authentication systems. Analog to digital converters take an analog input signal and produce a digital output stream, which is a numeric representation of the original analog signal. The analog biological feature is converted to a digital representation. Rather than use raw data from the sensor, biometric systems often process this data to extract only the information relevant to the authentication process. Further processing may be done in order to enhance differences and compress data. Once digital representation has been processed to the desired point, the digital representation is then stored; the stored digital biological feature is called a template. Most biometric devices will take multiple samples during the enrollment process to account for degrees of variance in the measurement of these features.

Once the user is enrolled, the biometric is used to verify the user's identity. When the user needs to be authenticated, the user's biological feature is acquired from the sensor. The sensor's analog information is converted to a digital representation. Then, this.digital representation is compared to a stored biometric template. The digital representation used for verification is called the live scan. The live scan typically does not exactly match the user's stored template. Since there are almost always variations in biometric measurements, these systems can not require an exact match between the user's original enrollment template and a current pattern. Instead, the current pattern is considered valid if it is within a certain statistical range of values. A comparison algorithm is used to determine if the user being verified is the same user as was enrolled.

The comparison algorithm yields a result of how close the digital representation is to the stored template. If the result falls into an "acceptable" range, an affirmative response is given; if the result falls into an "unacceptable" range, a negative response is given. The "acceptable" differs for each biometric. For some biometrics, the system administrator may set the level of the acceptable range. If this level is set too low, the biometric fails to be a valid authentication mechanism. If this level is set too high, the authorized users may have trouble being authenticated. This pattern matching is fundamental to the operation of any biometric system, and therefore should be considered a primary factor when evaluating a specific biometric product.

In general, most available biometric authentication mechanisms function as is stated in the above paragraphs. One key feature of biometrics is the template. The accumulated templates of all users are referred to as the template data base. Each system will require separate template databases for its authorized users. This database will require the same protections as the password databases. For each biometric system the size of the templates will vary. When testing these systems for accuracy, templates should be examined to determine if unique biometric features are adequately represented.

Another aspect of templates that affects the biometric authentication is the template retrieval for the comparison algorithm. The template may be used for identification or verification of users. Most devices use a verify, but some use identify. A biometric identify will take a live scan and compare it against the entire template data base to determine if any fall within the acceptable comparison algorithm range. A biometric verify will only compare a single user's template based on who the user claims to be. For example, a user will type in a user name and then take a live scan for verification. The comparison algorithm will only use the template used for that user name. Verification biometrics are typically faster because they do not have to compare the live scan against the entire template database.

5.3 Recommendations

When choosing a biometric authentication system, performance should be of importance. The performance of biometric authentication systems can be categorized by two measures, the False Acceptance Rate (FAR) and the False Rejection Rate (FRR) [38]. The FAR, also called type 2 errors, represents the percentage of unauthorized users who are incorrectly identified as valid users. The FRR, also called type 1 errors, represents the percentage of authorized users who are incorrectly rejected. The levels set in the comparison algorithm have a direct effect on these rates. How these rates are determined is fundamental to the operation of any biometric system, and therefore should be considered a primary factor when evaluating a biometric system. Some caution should be given to the FAR and FRR numbers from manufacturers because these numbers are extrapolated from small user sets and the assumptions for the extrapolations are sometimes erroneous. The physiological biometrics tend to have a better false acceptance rate because of the stability of the measured characteristic and because a behavioral characteristic is more likely able to be duplicated by other users.

These performance factors should be coupled with the type of users that will use the biometric. Some user factors may include learning curve and alternate access for those who may not be able to use the biometric. For each device the user must become familiar with the device for proper live scans to be taken. A nominal time that users take before the false rejection rate drops off is two weeks. Another user consideration is that not all users may be able to use the biometric. A user may have an impairment which prevents them from taking an acceptable scan. An alternate method is needed to grant those users access, or a biometric should be selected based on the needs of each set of users. When selecting a biometric, user acceptance should also be considered. Some biometrics have met with resistance from users because they are too invasive.

An ideal biometric is a non-invasive biometric with continuous authentication. In other words, the user does not need to take any additional action to be authenticated, and because it is non-invasive, the live scan may be done continuously. The continuous authentication will ensure another individual is not allowed access after an individual authenticated for access. Video facial scans and typing pattern biometrics are techniques which lend themselves to continuous authentication.

Once the type of biometric authentication mechanism has been established, the authentication mechanism must be attached to the access mechanism in the system. Typically, the sensor is an external hardware box with the analog to digital converter in it. The data compression and comparison algorithm is implemented with a combination of hardware and software. The path between the comparison algorithm to the access mechanism must be a trusted path. The output of most comparison algorithms is a pass or fail response which may be duplicated if the path is available. Also note if the sensor is shared for access to several systems, each system should have its own comparison algorithm and template data base.

5.4 Example

The example system for this section will be based on fingerprint biometrics, in combination with smartcards. Each system user is required to go through an enrollment process, during which the user's digitized fingerprint template is obtained. This template data is compressed and stored on a smartcard, which is issued to the user. In addition, a cryptographic authentication protocol is used to prove the identity of the smartcard to host computer systems and vice-versa. The smartcard therefore must share a cryptographic key with each host system that the user is allowed to access. These keys are loaded onto the smartcard during the enrollment process at the same time the fingerprint template is loaded. The fingerprint scanning hardware is built into the smartcard reader, so that only one device is required to communicate with the card and acquire the live fingerprint scan. At the end of the enrollment process, the user is given a smartcard containing the encrypted fingerprint template and cryptographic keys.

When a user wishes to log onto the system, the user inserts their smartcard into a reader/writer attached to a workstation. The user then provides a live fingerprint scan through the scanning mechanism built into the reader/writer. The reader/writer sends the live scan to the smartcard, which compares it to the template stored during enrollment. If the comparison is successful, the smartcard engages the workstation in a cryptographic handshake using the key it shares with the workstation. An alternative to the cryptographic handshake would be for the card to transmit a straightforward positive signal to the workstation. However, if the smartcard transmitted a simple positive/negative response to the workstation in place of the cryptographic handshake, an attacker might be able to duplicate the positive response and gain unauthorized access to the system. Another alternative would be to encrypt the current date and time and transmit this value to the workstation. No two encryptions should contain the same date and time, and so playback attempts could be easily detected.

Although the fingerprint biometric is the primary authentication mechanism in this example system, the use of a smartcard to store the enrollment template eliminates the need to store templates in databases on host systems. Users can carry templates with them by carrying the cards. In addition, templates do not need to be transmitted across network pathways because the comparison is done locally. The cryptographic keys which the cards and host systems use for authentication must be distributed when cards are set up during enrollment, but once the keying relationships have been established no sensitive information is exchanged between the cards and host systems during the login process. In contrast, live scans would need to be transmitted to a host system for each authentication if the comparisons were not done locally by the smartcard or reader/writer. This would increase the opportunities for attacks which involve interception and playback of live scans, although the data could be encrypted to reduce this risk. The approach described in this example has the following advantages: the enrollment database is distributed onto smartcards which each system user can carry; the distance which the live scan must travel between the acquisition hardware and the comparison process is short; and the authentication response between the card and a host system is cryptographically secure.

This example authentication system makes the assumption that the smartcard has sufficient processing power to compare a live fingerprint scan to the stored template. Existing smartcard architectures would have difficulty supporting a computationally intensive biometric comparison algorithm. The comparison process could be implemented more easily in the reader/writer using current technology. However, the approach chosen for this example is intended to illustrate the security advantages of including both template storage and the comparison algorithm on a smartcard. As smartcard technology progresses and biometric algorithms are improved, this approach should become more practical. It may even be possible to build the live scan sensors into the smartcard in the future.

6. COMBINATION METHODS

Passwords, authentication tokens, and biometrics are subject to a variety of attacks. Passwords can be guessed, tokens can be stolen, and even biometrics have certain vulnerabilities. These threats can be reduced by applying sound design principles and system management techniques during the development and operation of an authentication system. One method which can substantially increase the security of an authentication system is to use a combination of authentication techniques.

For example, an authentication system might require users to present an authentication token and also enter a password. By stealing a user's token, an attacker would still not be able to gain access to the host system, because the system would require the user's password in addition to the token. Although it might be possible to guess the user's password, the host system can make this extremely difficult by locking the user out after a specified number of invalid passwords have been presented in succession. Once a user's account has been locked in this manner, only the appropriate system administrator or security officer should be able to unlock the account.

Tokens can also be used to store biometric templates for user authentication. After enrollment, the user's unique template could be stored on a token, rather than in a file on the host system. When the user requests access to the system, a current template would be generated and compared to the enrollment template stored on the user's token. It would be preferable for this comparison to be carried out internally by the token, as in the example of Sec. 5, because the enrollment template would never need to leave the token. However, this is often not possible due to the complexity of the algorithms used for the comparison. The microprocessors typically used in smart tokens are not capable of executing these algorithms in a reasonable period of time. If the template comparison is done by the host system, the host must provide adequate assurance that user's templates cannot be compromised. In addition, the token and host system should implement an authentication protocol which assures that the host system is obtaining the template from a valid token, and that the token is submitting the template to a valid host. The ideal situation would be to have both the biometric sensors and the comparison algorithm implemented on the token. If this were the case, the token could perform the entire biometric authentication process. Technological advances should make it possible to realize this goal in the future.

7. CRYPTOGRAPHY IN AUTHENTICATION SYSTEMS

7.1 Overview

Cryptography is the process of scrambling information in such a manner that it becomes unintelligible, and can only be unscrambled by the intended recipient(s). In cryptographic terms, this process involves the encryption of plaintext data to produce ciphertext, and the subsequent decryption of ciphertext to recover the original plaintext. Encryption and decryption are therefore inverse processes. Cryptographic processing depends on the use of keys, which are of primary importance in the security of a cryptographic system. Cryptographic keys are conceptually similar to the keys used with padlocks, in the sense that data can be locked, or encrypted, through the use of a key in conjunction with a cryptographic algorithm. Symmetric key algorithms decrypt data with the same key used for encryption [26]. Asymmetric key algorithms use a pair of keys, consisting of a public key component and a private key component, which have a specific mathematical relationship [24]. Symmetric and asymmetric key algorithms are commonly referred to as secret key and public key algorithms, respectively. Cryptography plays a major role in information security, and is a critical component of authentication technology.

7.2 Secret Key Cryptography

The primary feature which distinguishes secret key algorithms is the use of a single secret key for cryptographic processing. Secret key cryptography has been in use for thousands of years in a variety of forms. Modern implementations usually take the form of algorithms which are executed by computer systems in hardware, firmware or software. The majority of secret key algorithms are based on operations which can be performed very efficiently by digital computing systems. In 1977, the National Institute of Standards and Technology developed Federal Information Processing Standards Publication 46, which describes the DES secret key cryptographic algorithm [26,39,40]. The use of DES is mandated for applications within the federal government which require cryptographic processing of sensitive unclassified information. The DES algorithm has been implemented in a wide variety of commercial products, including many which deal with authentication. This algorithm can be implemented with reasonable efficiency in the firmware of a smart token, and a number of DES based smart tokens are commercially available.

There are a variety of authentication protocols which rely on cryptography. One of the most fundamental of these is the challenge-response protocol [23], which can be readily implemented between a host computer system and a smart token or biometric device, or another computer system. As an example, assume that a host computer system shares a secret cryptographic key with an authentication device, such as a smart token. The host system and authentication device both have cryptographic capabilities, each using the shared secret key for encryption and decryption. The challenge-response protocol could proceed as follows:

1. The host system generates a random number RN1 and transmits

this number to the authentication device.

2. The authentication device encrypts RN1 and generates a second random number RN2. The encrypted value of RN1 and the plaintext value of RN2 are sent back to the host system.

3. The host decrypts RN1 and compares the resulting plaintext to the value transmitted in step 1. If the two values match, the host system is satisfied that the authentication device is in possession of the correct secret key, and hence the identity of the authentication device is verified.

4. The host encrypts RN2 and transmits this value to the authentication device.The authentication device then decrypts RN2 and compares the plaintext value to the original value for RN2 transmitted in step 2. If the two values match, the authentication device accepts the claimed identity of the host system.

There are several aspects to this type of protocol which affect the security of an authentication system. The protocol is dependent on the possession of shared secret keys for proof of identity. The protection provided for these keys is of utmost importance, since an unauthorized party who obtained a valid key could perform the encryption required during the authentication process and thereby pose as a legitimate user.

The challenge-response protocol provides mutual authentication, meaning that the identity of each party involved in the authentication process is verified. Computer users have traditionally been required to provide proof of identity to host computer systems, with little or no assurance that the claimed identity of the host system is correct. This situation leads to the possibility that a user could be spoofed by a bogus host computer which mimics a legitimate computer. The user's password and possibly other sensitive information could be collected by the bogus computer, since the user has no mechanism for verifying the identity of the host computer systems. Mutual authentication protocols provide users with the ability to differentiate between legitimate hosts and computers which are attempting to spoof users.

7.3 Public Key Cryptography

Recent advances in cryptographic technology have led to the development of public key cryptographic algorithms. These algorithms are referred to as "asymmetric", because they rely on two different keys to perform cryptographic processing of data. These keys are generated and used in pairs consisting of private and public key components.

Public key cryptosystems make possible authentication schemes in which a secret can be verified without the need to share that secret. In public key cryptography each user independently generates two mathematically related keys. One is typically made public, so it is referred to as the public key. The other is kept private so it is referred to as the user's private key. The public key becomes in effect part of the user's identity, and should be made as well known as necessary, like a phone number. Conversely, the private key should be known only to the user, since it can be used to prove ownership of the public key and thus the user's identity. It is computationally infeasible to derive a user's private key from the corresponding public key, so free distribution of the public key poses no threat to the secrecy of the private key.

The private key can be used to create what are known as digital signatures [24]. Similar to a written signature, a digital signature is unique to the signer except that it is much more difficult to forge, and can be verified electronically. This is made possible by the fact that in public key cryptosystems, digital signatures are generated with the private key component of the public/private key pair. The corresponding public key is used to verify the signature. Since a given user's private key does not need to be shared with other parties, there is a strong association between the user's identity and possession of the private key. Digital signatures can be used for authentication as follows: when a host system wishes to verify the identity of a user who is in possession of a particular private key, the host system can challenge the user with an electronic message. The user would sign this message with the private key and return the signature to the host system. The host can then verify the signature, and thus the identity of the user, with the user's public key. Since only one specific user possesses a particular private key, a signature generated by this key is strong proof of the user's identity. These authentication messages usually contain a time variant parameter, to prevent replay of old messages. This approach requires that the host system have access to the public key of the user from a trusted source. If system users have access to signature verification capability, mutual authentication protocols can be supported. The security of authentication protocols based on public key cryptography is dependent on the level of protection provided for private keys, and the degree to which a verifier trusts the source of public keys. The CCITT X.500 recommendation describes one approach to the design of a directory service for the certification and management of public keys [41].

Public key cryptography can in theory be more convenient for authentication than secret key cryptography since it is not necessary for two parties wishing to authenticate each other to share a secret key. Hence, a less complicated key distribution system may be required. Also, public key cryptography makes it possible to place the authentication information under the direct control of the system user. For access control, this is especially helpful since secret authentication information need not be distributed throughout the system. However, public key systems generally require arithmetic operations which are difficult for small microprocessors. This can cause problems in the design of authentication systems since it is often necessary for the cryptographic algorithms to be implemented on a small device with limited processing power. It can be very difficult to obtain satisfactory performance from a smart card or reader/writer in terms of public key operations. However, this deficiency can be compensated for to some extent by distributing operations between the authentication system and the host computer system. In addition, advances in integrated circuit technology are increasing the capabilities of devices such as smart cards to the point where an acceptable level of performance can be attained in the implementation of public key algorithms. Hybrid approaches are also possible, where public key cryptography is used to distribute keys for use by secret key algorithms. NIST Special Publication 800-2 [42] provides a comprehensive survey of public key technology.

7.4 Cryptographic Authentication Protocols

Cryptographic algorithms and the devices which implement them are important components of many authentication systems. In order for these components to work together effectively to accomplish the authentication process, protocols must be established to specify how the cryptographic algorithms will be used [1]. A variety of cryptographic authentication protocols have been developed by members of the private sector, the federal government, and the academic community. Since any computer system which uses cryptographic authentication will require one or more of these protocols, it is useful to compare two representatives which are well documented and appear to be gaining widespread support. The Kerberos authentication system, developed by the Massachusetts Institute of Technology, is based on secret key cryptography. Digital Equipment Corporation has produced a distributed authentication service known as SPX, utilizing public key cryptography. A short description of these protocols is included in this document for those wishing to understand the issues involved in the design and selection of authentication protocols in general, and in no way implies an endorsement by the National Institute of Standards and Technology. Also, these protocols are continually evolving and the reader should consult current references for an accurate description of the most recent version of these protocols. The Distributed Authentication Security Service (DASS) which serves as the basis for the SPX implementation has become a series of Internet Requests For Comment (RFCs) [43,44,45,46], and is no longer supported by Digital Equipment Corporation as a product. Kerberos Version 5 corrects many of the deficiencies of Version 4, and has also been accepted as Internet RFC 1510 [47]. Kerberos Version 4 is discussed in this section.

Kerberos and SPX are similar in a number of ways. Both protocols are designed to reduce the risk of impersonation in an environment of networked computer systems. Each protocol relies on cryptographic techniques to provide strong authentication of users and host computer systems. Further, Kerberos and SPX both use a trusted third party to manage the cryptographic keying relationships which are critical to the authentication process. System users have a significant degree of control over the workstations which are used to access network services, and these workstations must therefore be considered untrusted. The following discussion focuses on the major characteristics of Kerberos and SPX, and should not be considered a complete technical definition of either protocol. For example, delegation of authority and authentication which crosses the boundaries of multiple key distribution centers are not discussed.

7.4.1 Kerberos

Kerberos was developed to provide distributed network authentication services at MIT [48]. In this environment, users access a network of computing resources through workstations which are assumed to be untrusted. An unscrupulous user might therefore be able to subvert a given workstation with relatively little difficulty. A primary threat in this type of client-server system is the possibility that one user will be able to claim the identity of another user, thereby gaining access to system services without the proper authorization. To protect against this threat, Kerberos provides a trusted third party accessible to network entities, which supports the services required for authentication between these entities. This trusted third party is known as the Kerberos key distribution server, which shares secret cryptographic keys with each client and server within a particular realm.

The Kerberos authentication model is based upon the presentation of cryptographic tickets to prove the identity of clients requesting services from a host system, or server [48]. Since a computer workstation typically performs the client side of an authentication protocol on behalf of a human user, the term "client" in the following discussion will refer to a specific system user and the computer workstation associated with that user. A summary of the Kerberos authentication process follows (see also Figure 4):

1. The user initiates a login process on a workstation. The workstation login process transmits a ticket request to the key distribution server. The ticket request contains the client identity, the identity of the target service, and the current time.

2. The key distribution server retrieves the secret keys for the client and the service. A ticket is prepared, consisting of: a temporary session key for use by the client and the service, the client identity, the service identity, a timestamp, the workstation address, and the validity interval of the ticket. The ticket is then encrypted under the key of the service. The encrypted ticket, temporary session key, identity of the service, validity interval, and timestamp are encrypted under the client's secret key.

3. The encrypted response packet is sent from the key distribution service to the client.

4. When the client receives the encrypted response packet, the client's secret key is generated by performing a one-way encryption of the user's password. Using this key, the client decrypts the response packet and verifies the origin of the message based on the timestamp and client identity. The client then creates an authenticator consisting of the client identity, client address, and a timestamp. This authenticator is encrypted under the temporary session key.

5. The client sends the authenticator and the ticket obtained in step 4 to the service.

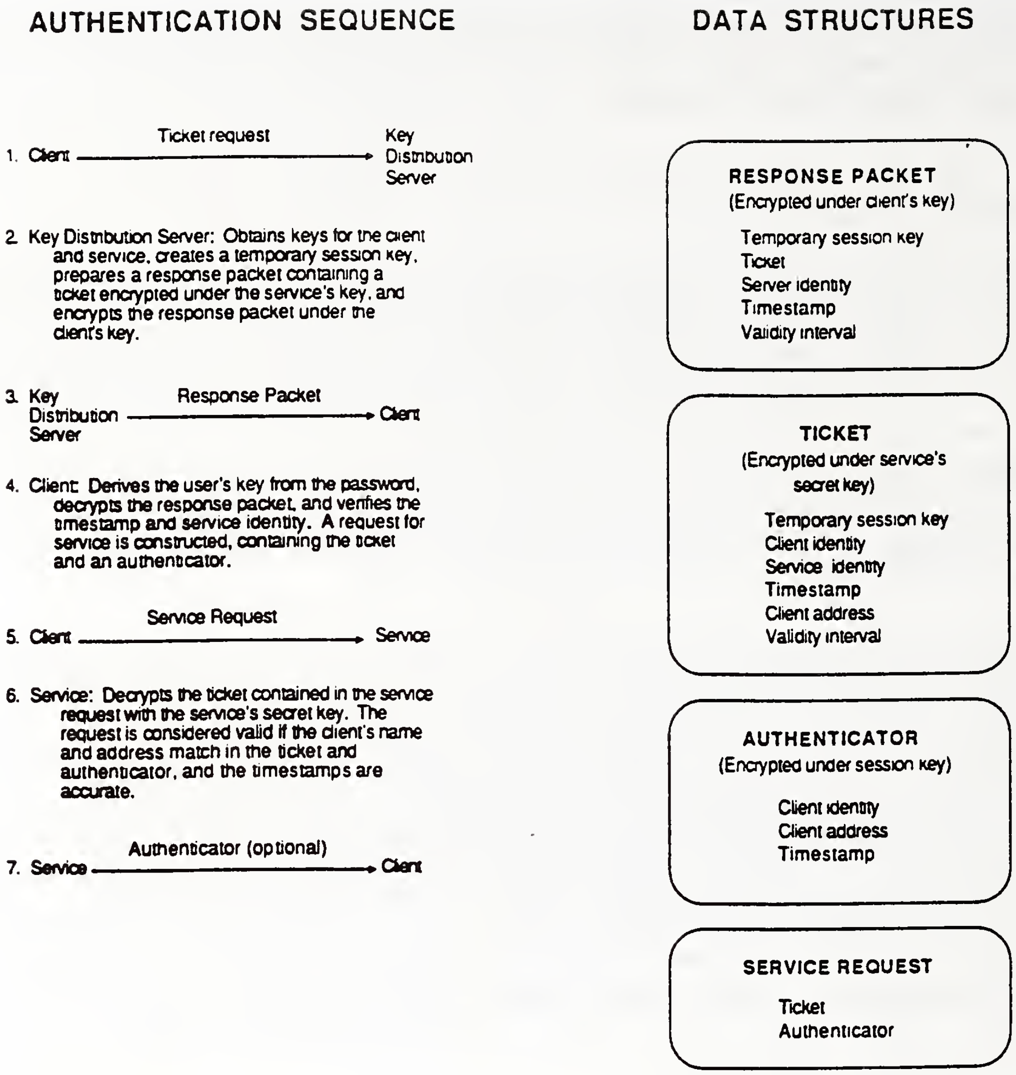
6. The requested service decrypts the ticket using its secret key and verifies the contents of the ticket, proving that the ticket originated from the key distribution server. The temporary session key is obtained from the ticket and used to decrypt the authenticator. The in-

formation stored in the authenticator proves the identity of the client to the requested service, which can then respond to the service request in the appropriate manner.

7. The service may optionally return an authenticator to prove its identity to the client.

Although a detailed analysis of the Kerberos protocol is beyond the scope of this document, several points are worth noting. Kerberos eliminates the need for each client to share a unique authentication key with each service, placing the responsibility for managing keying relationships on the Kerberos key distribution center. Use of a key distribution/key translation center as a trusted third party in secret key authentication protocols is a fairly common technique for reducing the total number of keying relationships which must be managed [49]. However, the key distribution server requires a very high level of protection, since an attacker could theoretically gain access to keys for all clients and servers within a given realm by compromising the key distribution server for that realm.

Since a user's secret key is a one way function of the user's password, these keys are subject to many of the same attacks used against password based authentication systems which apply a one way function to user's passwords before storing them. Anyone can request a ticket from the key distribution service, so an attacker could obtain a ticket for any given user by transmitting the client name, service name, and timestamp to the key distribution service. The attacker could then attempt to guess the user's password, and thus the user's secret key, offline. It would be easy to tell when the correct password had been guessed, because the resulting secret key would decrypt the encrypted ticket obtained from the key distribution service correctly. Possession of a user's password would then allow the attacker to pose as that user. Good password management techniques are essential to minimize the risk of such attacks.



AUTHENTICATION SEQUENCE

1. Client —Ticket request→ Key Distribution Server

2. Key Distribution Server: Obtains keys for the client and service, creates a temporary session key, prepares a response packet containing a ticket encrypted under the service's key, and encrypts the response packet under the client's key.

3. Key Distribution Server —Response Packet→ Client

4. Client: Derives the user's key from the password, decrypts the response packet, and verifies the timestamp and service identity. A request for service is constructed, containing the ticket and an authenticator.

5. Client —Service Request→ Service

6. Service: Decrypts the ticket contained in the service request with the service's secret key. The request is considered valid if the client's name and address match in the ticket and authenticator, and the timestamps are accurate.

7. Service —Authenticator (optional)→ Client

DATA STRUCTURES

RESPONSE PACKET
(Encrypted under client's key)
Temporary session key
Ticket
Server identity
Timestamp
Validity interval

TICKET
(Encrypted under service's secret key)
Temporary session key
Client identity
Service identity
Timestamp
Client address
Validity interval

AUTHENTICATOR
(Encrypted under session key)
Client identity
Client address
Timestamp

SERVICE REQUEST
Ticket
Authenticator

Figure 4: Authentication sequence and data structures for Kerberos

7.4.2 SPX

SPX is an implementation of the Distributed Authentication Security Service (DASS) [44], an authentication architecture based on public key cryptography. As in Kerberos, the primary goal of SPX is to prevent system users from claiming the identity of other users in a distributed computing environment. The SPX architecture includes an Application Programming Interface, known as the Generic Security Services Application Programming Interface (GSS-API) [45,46].

A certificate distribution center is used to distribute public key certificates. User enrollment services are provided by the login enrollment agent facility, which generates and distributes encrypted authentication data.

The functions of a certificate distribution center in SPX are similar to those of the Kerberos key distribution center. In both systems, users are required to register cryptographic keys with a trusted third party, so that network entities can access these keys as required during authentication exchanges. However, public key cryptography provides some advantages in the key management process of SPX. Since the keys stored in the certificate distribution center are public, disclosure is not a threat. If an attacker were to obtain the public key for a particular client from the certificate distribution center, it would not allow the attacker to pose as the client. The attacker would need to compromise the private key of the client, or generate a new public/private key pair and replace the client's registered public key with the new public key without being detected. The SPX authentication sequence follows (see also Figure 5):

1. A client requests a public key certificate for a service from the certificate distribution center. The certificate distribution center returns the certificate, which binds the identity of the service to the public key of that service.

2. The client verifies the certificate with the public key of a trusted certification authority. The client then generates a DES authentication key for this session, and creates a ticket consisting of a delegation public key and a validity interval. The ticket is signed by the client's private key. An authentication token is then constructed from the client's identity, the ticket, the DES session key encrypted under the service's public key, a signature on the encrypted DES key, and an authenticator consisting of a timestamp and cryptographic checksum.

3. The client's authentication token is transmitted to the service.

4. The service uses its private key to decrypt the DES session key stored in the token, and then uses this DES key to verify the timestamp within the authenticator. Verification of the timestamp proves that the token is valid within the limits set for the lifetime of authentcation tokens.

5. The service requests a certificate for the client from the certificate distribution center.

6. The validity of the token is accepted at this point, but the association between the client who sent the token and the identity claimed by that client has not yet been proven. The server extracts the client's public key from the certificate obtained in step 5, and verifies the signature associated with the ticket. If the signature is valid, the service accepts the identity of the client.

7. Mutual authentication is an option in the SPX architecture. In situations where there is a need for the service to authenticate to the

client, the DES session key obtained from the client's authentication token in step 1 is used to calculate a cryptographic checksum on a timestamp, creating a new authenticator. This authenticator is returned to the client, who verifies the authenticator based on the validity of the cryptographic checksum.

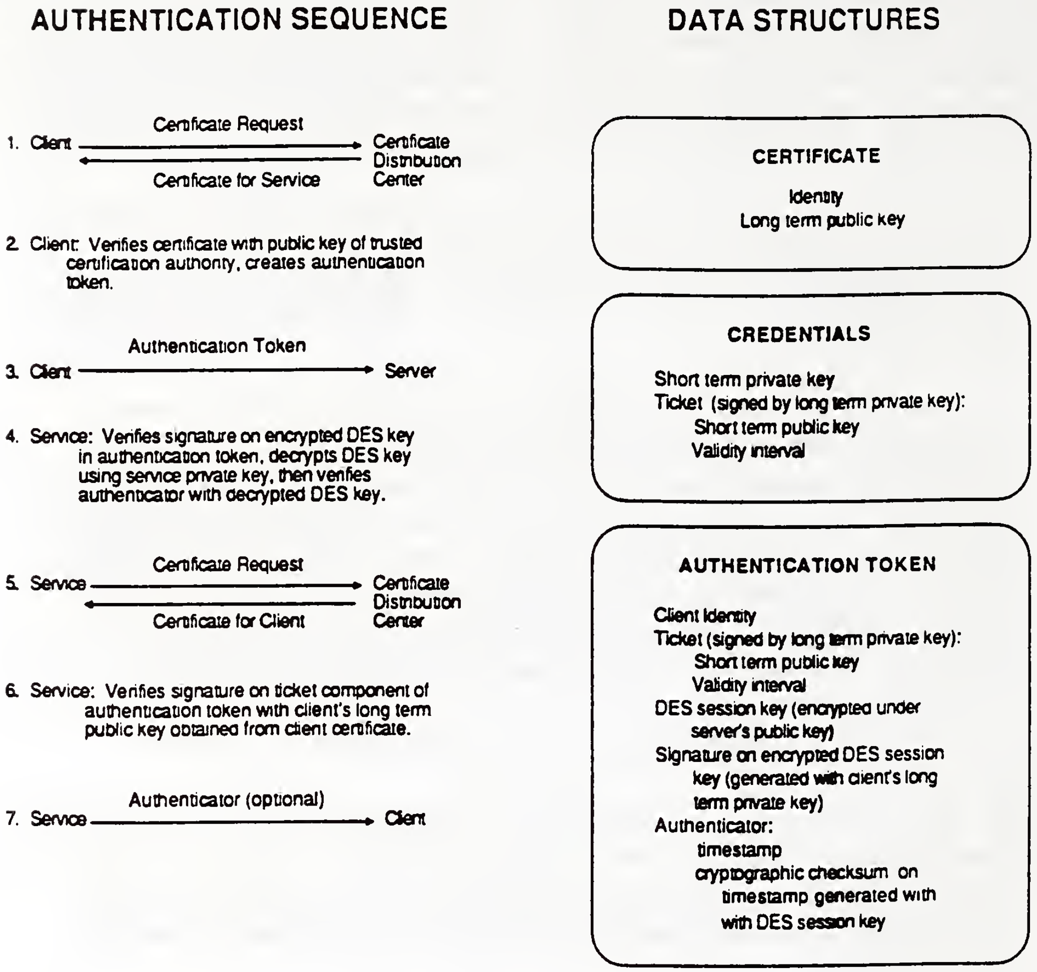

AUTHENTICATION SEQUENCE
DATA STRUCTURES
1. Client
Certificate Request
Certificate Distribution Center
Certificate for Service
2. Client: Verifies certificate with public key of trusted certification authority, creates authentication token.
3. Client
Authentication Token
Server
4. Service: Verifies signature on encrypted DES key in authentication token, decrypts DES key using service private key, then verifies authenticator with decrypted DES key.
5. Service
Certificate Request
Certificate Distribution Center
Certificate for Client
6. Service: Verifies signature on ticket component of authentication token with client's long term public key obtained from client certificate.
7. Service
Authenticator (optional)
Client
CERTIFICATE
Identity
Long term public key
CREDENTIALS
Short term private key
Ticket (signed by long term private key):
Short term public key
Validity interval
AUTHENTICATION TOKEN
Client Identity
Ticket (signed by long term private key):
Short term public key
Validity interval
DES session key (encrypted under server's public key)
Signature on encrypted DES session key (generated with client's long term private key)
Authenticator:
timestamp
cryptographic checksum on timestamp generated with with DES session key

Figure 5: Authentication sequence and data structures for SPX

If the checksum is correct, the client knows that the checksum was generated with the original DES session key. Since this DES key was encrypted under the service's public key in the original authentication token, only the service's private key could have decrypted it.

The fundamental difference between Kerberos and SPX lies in the choice of secret key versus public key cryptography. Management of cryptographic keys is a primary issue in any system which relies on these keys for authentication. Both protocols place the responsibility for key management on a trusted third party. The role played by the certificate distribution center in SPX is similar to the Kerberos Key Distribution Center. However, public key cryptography provides some advantages in the key management process of SPX. Since the keys stored in the certificate distribution center are public, disclosure is not a threat. If an attacker were to obtain the public key for a particular client from the certificate distribution center, it would not allow the attacker to pose as the client. The client's public key can only be used to verify the client's signature, and so the attacker would need to compromise the private key of the client or generate a new public/private key pair and replace the client's registered public key with the new public key without being detected. Since Kerberos is based on secret key cryptography, compromise of a particular Kerberos Key Distribution Center key database means that the secret keys which the Center shares with each client have been compromised [50]. An attacker could then use these secret keys to assume the identity of any of the clients registered with the Center. Kerberos therefore requires a greater level of protection for the key database of the trusted third party than does SPX.

The security of the Kerberos and SPX protocols can be enhanced through the use of authentication tokens with cryptographic capabilities. If the cryptographic operations required by the authentication protocols are performed by smart tokens, workstations need not be trusted with all aspects of the cryptographic process. Providing adequate protection for cryptographic keys stored on workstations running untrusted operating systems is a difficult problem. Smart tokens can eliminate the need to store keys on client's workstations, since the keys can be kept on the tokens instead. Both protocols are compatible with smart token technology, and an implementation of the Kerberos system which utilizes a smartcard for cryptographic processes has been demonstrated [51]. Kerberos Version 5 specifically supports the use of authentication tokens as an option for stronger authentication [47].

8. GENERAL IMPLEMENTATION GUIDELINES

There are numerous factors which must be considered when an organization decides to implement an advanced authentication system. The following general guidelines are intended to assist those responsible for evaluating, procuring, and integrating these systems:

8.1 RISK ANALYSIS - A thorough analysis should be done to determine what parts of the system in question are vulnerable to attack, and to prioritize these vulnerabilities in terms of severity and likelihood.

8.2 PRODUCT EVALUATION AND SELECTION - Once the risks associated with a host system have been identified, this information can be used to develop the requirements for an authentication system. A cost-benefits analysis should be included in this process to ensure the acquisition of a product which will meet the organizational security requirements in a cost effective manner. the authentication system will have to meet several requirements in order to function effectively in a given environment. The organization responsible for selecting the authentication system should decide whether sufficient in-house expertise exists to evaluate the available options. In some cases it is more cost-effective to hire a consultant who is familiar with the available technology. Whether the evaluation is done in-house or by a consultant, the following items should be considered:

8.2.1 Resources - There are a variety of resources which should be consulted when evaluating authentication systems. Vendor product literature can be very helpful in describing specific details of product operation, and in understanding the range of products offered. There are several annual conferences devoted to computer security, network access control, and authentication technology. In addition to the papers presented at these conferences, there are usually large vendor exhibit halls and product forums. Many organizations, particularly those in the government sector, have published information on the selection and integration of advanced authentication technology. These publications are often the result of practical experiences gained during the implementation of these systems, and so can be particularly useful.

8.2.2 Integration into existing environment - This factor is discussed further in the next section, but is an important consideration when selecting a product. All other features of an authentication system may be irrelevant if the product cannot be integrated into the customer's computing environment.

8.2.3 Custom design - There are cases where a commercial product which meets the needs of an organization may not be available. In these cases, the organization may decide to do a custom design using in-house resources. This alternative would be most practical for large organizations with experienced system design and support groups, or for smaller organizations with a high level of expertise in computer access control systems. Vendors are often willing to work with customers to modify existing products or design new products to meet custom requirements. An arrangement which often works well is for the customer and vendor to work together on the design of the system, and for the vendor to then manufacture the product.

8.2.4 Cost and performance - The relationship between cost and performance can be relatively complex for authentication technology. Similar products from different vendors may vary widely in cost, depending on the vendor's manufacturing and development techniques and marketing philosophies. In general, it can be expected that devices with a higher performance level will cost more, but individual cases should be evaluated carefully. The general approach should be to procure the authentication system which provides the required level of security and other performance factors at a minimum cost.

8.2.5 Accuracy - The accuracy of an authentication system refers to the ability of that system to correctly identify authorized system clients while recognizing unauthorized clients. Since this is the primary function of an authentication system, accuracy is directly related to the level of security provided by the system. There are no widely accepted standards for evaluating the accuracy of the authentication process, and results published by vendors may not be objective. For these reasons, an organization should thoroughly understand the applicable test methodologies and run independent tests where necessary to determine the accuracy of an authentication system in terms which are relevant to the environment in which the system will be used.

8.2.6 Reliability - An authentication system should be capable of operating in its intended environment for a reasonable period of time. During this time period, the system is expected to perform at or above a level which insures an appropriate amount of protection for the host system. If the authentication system fails, it should be in such a way as to minimize the chances for unauthorized access during the failure.

8.2.7 Maintainability - All hardware and software systems require some form of maintenance. The components of an authentication system should be evaluated to determine the level of maintenance which the system will require. One of the goals in the design of an authentication system should be to minimize the maintenance requirements within the constraints of system cost, performance, and available technology.

8.2.8 Commercial availability - Large-scale networking of computer systems and distributed computing are relatively recent developments, and are the driving forces behind the need for more effective methods for authenticating system clients. Therefore, the market for advanced authentication technology is not fully developed and somewhat unstable. Because many commercially- available authentication systems have not yet been sold in quantity, an organization which is considering the use of this technology should evaluate the vendor's ability to produce systems in quantity which meet pertinent quality control standards. Contracts written to procure authentication systems should provide some form of protection for the customer in the event that the vendor is unable to produce systems in the quantities required.

8.2.9 Upgradeability - Because the technology of advanced authentication systems is continually developing, it is desirable for any authentication system to be able to accommodate the replacement of outdated components with new ones. A modular approach to the design of an authentication system, with clearly defined interfaces between the system components, will facilitate the process of upgrading to new technology.

8.2.10 Interoperability - A wide variety of computing platforms and security architectures are in use today. Any authentication system should be designed to work with as many of these diverse platforms as possible, or at least to require a minimum of modifications to work in different environments.

8.2.11 Reputation of manufacturer - Obtaining satisfactory service during the selection, installation, and long term operation of an authentication system can be difficult if the manufacturer is uncooperative. If the manufacturer goes out of business, other vendors may be unwilling to provide service for the original manufacturer's system, with which they are unfamiliar. Customers can request a list of references from prospective vendors for products and services which have been provided to other customers in the past. In addition, the resumes of key individuals working on the vendor's staff can sometimes be examined to determine whether an adequate level of expertise is available.

8.3 SYSTEM INTEGRATION - The integration of an authentication system into an existing computer environment can be very difficult. Few operating systems, if any, contain well-defined entry points for replacing the default authentication mechanism supplied with the operating system. This is partly because there is no widely accepted standard for the interface between an operating system and an authentication device. Until such a standard becomes available, there are three general options:

8.3.1 In some cases, the vendor who provides the authentication system may have already integrated it into certain operating systems. If the authentication system meets the requirements of the customer and the customer is using the specified operating system, then the system integration has already been accomplished.

8.3.2 Operating system vendors may select certain security architectures for incorporation into their systems. If these architectures include an authentication technology which the customer finds acceptable, then the operating system may be purchased with the appropriate authentication mechanism as part of the package.

8.3.3 Often, it will be necessary to customize the authentication system and perhaps modify the host operating system so that the two can communicate. This will involve cooperation between the operating system vendor, the authentication system vendor, and the customer, unless the customer has sufficient expertise to perform the integration in-house. A prototyping approach is strongly recommended, due to the complexity of this type of project. Implementing such a system on a small scale first can be very helpful in determining what problems will be encountered in a full-scale implementation.

8.4 SYSTEM MAINTENANCE - After an authentication system has been selected and installed, it must be maintained. Long term plans for system maintenance should be developed by the customer or provided by the vendor long before the system is installed, because the cost of maintaining a system can easily exceed the initial acquisition cost if the system is to be in operation for a reasonable length of time. Provisions must be made for assigning responsibilities for system administration so that new clients can be enrolled, inactive accounts can be deleted, and system malfunctions can be identified and corrected.

The majority of authentication systems employ cryptography, which means that some form of cryptographic key management system will be necessary. The key management component may be provided by the authentication system vendor, but the process of maintaining and distributing keys usually requires active participation by the host system. Since the security of a cryptographic system is directly related to the level of protection provided for the cryptographic keys, it is essential for the vendor or customer to develop a system for managing these keys effectively. Also, the host computer system will probably evolve over time through the addition of new software and hardware, and these changes may require corresponding modifications or upgrades to the authentication system to maintain compatibility.

9. CONCLUSION

As the nation's dependence on computer services grows, so will the number and complexity of the computer networks which serve this need. Modern information processing systems are interconnected to a greater degree than ever before. This high degree of interconnection poses some unique challenges to those responsible for the security of these systems. It is difficult to maintain the accessibility of open computing environments while protecting these systems from malicious or accidental misuse. Strong authentication is the first line of defense against these threats, because a system must be able to differentiate between authorized and unauthorized access attempts. Modern authentication technology provides solutions which are secure, convenient, and cost effective. The primary methods available today include passwords, authentication tokens, and biometrics. Password-only authentication may be adequate in some limited situations, but is often not suitable when used alone.

REFERENCES

1. Woo, T. Y. C., and S. S. Lam, Authentication for Distributed Systems, IEEE CS Press, January 1992.

2. Guideline on User Authentication Techniques for Computer Network Access Control, National Institute of Standards and Technology, Federal Information Processing Standards Publication 83, National Technical Information Service, Springfield, VA, September 1980.

3. Password Usage, National Institute of Standards and Technology, Federal Information Processing Standards Publication 112, National Technical Information Service, Springfield, VA, May 1985.

4. Automated Password Generator (APG), National Institute of Standards and Technology, Federal Information Processing Standards Publication 181, National Technical Information Service, Springfield, VA, March 29, 1994.

5. Smart Card Technology: New Methods for Computer Access Control, National Institute of Standards and Technology, NIST Special Publication 500-157, National Technical Information Service, Springfield, VA, September 1988.

6. Dray, J. F., M. E. Smid and R. Warnar, A Token Based Access Control System for Computer Networks, Proceedings - The 12th National Computer Security Conference, October 1989.

7. NIST SACS Reader/Writer Specification, Datakey, Inc., Report #065-0098-000, July 11, 1991.

8. ASACS Portable Reader/Writer Specification, Datakey, Inc., Report #065-0131-000, April 24, 1992.

9. Identification Cards - Integrated Circuit(s) Cards with Contacts - Part 2: Dimensions and Location of the Contacts, International Organization for Standardization, International Standard 7816-2, 1988.

10. Identification Cards - Contactless Integrated Circuit(s) Cards - Part 3: Electronic Signals and Transmission Protocols, International Organization for Standardization, International Standard 7816-3, 1989.

11. Dodson, D. F., J. F. Dray, and R. Warnar, "Security Features for an FMS Smart Card", National Institute of Standards and Technology Special Report, September 25, 1990.

12. Dray, J. F., and D. M. Balenson, An Overview of the Advanced Smartcard Access Control System (ASACS), Proceedings of the PSRG Workshop on Network and Distributed System Security, pp. 125-133, February 1993.

13. ASACS Smartcard Specification, Datakey, Inc., Report #065-0130-000, April 24, 1992.

14. Smartcard Application Program Interface for the Advanced Smartcard Access Control System (ASACS), TISR #465D, Trusted Information Systems, Inc., Glenwood, MD, October 1992.

15. Advanced Smartcard Access Control System (ASACS): Smartcard Command Set Interface, National Institute of Standards and Technology, U.S. Department of Commerce, Washington, D.C., 1992.

16. Advanced Smartcard Access Control System (ASACS): Reader/Writer Command Set Interface, National Institute of Standards and Technology, U.S. Department of Commerce, Washington, D.C., 1992.

17. Security Officer Maintenance (SOMAINT) Program User's Manual, National Institute of Standards and Technology, U.S. Department of Commerce, Washington, D.C., 1992.

18. Advanced Smartcard Access Control System (ASACS): The DSS Signature Utility Program Manual, National Institute of Standards and Technology, U.S. Department of Commerce, Washington, D.C., 1992.

19. Advanced Smartcard Access Control System (ASACS): UNIX Access Control Software Manual, National Institute of Standards and Technology, U.S. Department of Commerce, Washington, D.C., 1992.

20. NIST SACS Smartcard Specification, Datakey, Inc., Report #065-0097-000, July 11, 1991.

21. Hitachi H8/310 Single-Chip Microcomputer, Hitachi, Ltd., Tokyo, Japan, 1989.

22. Haykin, M. E., and R. Warnar, Smart Card Technology: New Methods for Computer Access Control, National Institute of Standards and Technology, NIST Special Publication 500-157, National Technical Information Service, Springfield, VA, , September 1988.

23. American National Standard X9.26-1990, Financial Institution Sign-on Authentication for Wholesale Financial Systems, American Bankers

Association, Washington, D.C., 1990.

24. Proposed Digital Signature Standard (DSS), National Institute of Standards and Technology, U.S. Department of Commerce, Washington, D.C., August 30, 1991.

25. Rivest, R. L., A. Shamir and L. M. Adleman, A Method for Obtaining Digital Signatures and Public Key Cryptosystems, Communications of the ACM, Volume 21, Number 2, February 1978, pp. 120-126.

26. Data Encryption Standard (DES), National Institute of Standards and Technology, Federal Information Processing Standards Publication 46-2, National Technical Information Service, Springfield, VA, Reaffirmed December 30, 1993 (Supersedes FIPS PUB 46, January 15, 1977).

27. International Standard 7816-1, Identification Cards - Integrated Circuit(s) Cards with Contacts -- Part 1: Physical Characteristics, International Organization for Standardization, 1987.

28. Kernigan, B. and D. Ritchie, The C Programming Language, 2nd Edition, Prentice Hall, 1988.

29. Secure Hash Standard (SHS), National Institute of Standards and Technology, Federal Information Processing Standards Publication 180, National Technical Information Service, Springfield, VA, May 11, 1993.

30. Linn, J., Privacy Enhancement for Internet Electronic Mail: Part I -- Message Encipherment and Authentication Procedures, Internet Request For Comments (RFC) 1421, July 23, 1992.

31. Kent, S., Privacy Enhancement for Internet Electronic Mail: Part II -- Certificate-Based Key Management, Internet Request For Comments (RFC) 1422, BBN Communications, February 1993.

32. Balenson, D. M., Privacy Enhancement for Internet Electronic Mail: Part III -- Algorithms, Modes, and Identifiers, Internet Request For Comments (RFC) 1423 , Trusted Information Systems, February 1993.

33. Kaliski, B., Privacy Enhancement for Internet Electronic Mail: Part IV -- Key certification and Related Services, Internet Request For Comments (RFC) 1424, RSA Laboratories, February 1993.

34. Kaliski, B., The MD2 Message-Digest Algorithm, Internet Request for Comments (RFC) 1319, RSA Laboratories, April 1992.

35. Rivest, R., The MD5 Message-Digest Algorithm, Internet Request for Comments (RFC) 1321, MIT Laboratory for Computer Science and RSA Laboratories, April 1992.

36. CCITT Recommendation X.509, The Directory - Authentication Framework, The International Telegraph and Telephone Consultative Committee, November 1988.

37. Galvin, J., et al, Security Issues of a UNIX PEM Implementation, TISR #468D, Trusted Information Systems, February 11, 1993.

38. Biometric Access Control Device Evaluation Criteria (Draft Report), DCI Intelligence Information Handling Committee, February 1991.

39. DES Modes of Operation, National Institute of Standards and Technology, Federal Information Processing Standards Publication 81, National Technical Information Service, Springfield, VA, December 2, 1980.

40. Guidelines for Implementing and Using the NBS Data Encryption Standard, National Institute of Standards and Technology, Federal Information Processing Standards Publication 74, National Technical Information Service, Springfield, VA, April 1, 1981.

41. X.500 Directory Services Recommendation

42. Public Key Cryptography, National Institute of Standards and Technology, NIST Special Publication 800-2, National Technical Information Service, Springfield, VA, April 1991.

43. Linn, J., Common Authentication Technology Overview, Internet Request For Comments (RFC) 1511, Geer-Zolot Associates, September 1993.

44. Kaufman, C., DASS: Distributed Authentication Security Service, Internet Request For Comments (RFC) 1507, Digital Equipment Corporation, September 1993.

45. Linn, J., Generic Security Services Applications Program Interface, Internet Request For Comments (RFC) 1508, Geer-Zolot Associates, September 1993.

46. Wray, J., Generic Security Service API: C-Bindings, Internet Request For Comments (RFC) 1509, Digital Equipment Corporation, September 1993.

47. Kohl, J. (Digital Equipment Corporation), and Neumann, C. (University of Southern California/Information Sciences Institute), The Kerberos

Network Authentication Service (V5), Internet Request For Comments (RFC) 1510, September 1993.

48. Steiner, J. G., C. Neuman, and J. I. Schiller, Kerberos: An Authentication Service for Open Network Systems, Proceedings of the Winter USENIX Conference, Dallas, Texas, March 30, 1988.

49. American National Standard X9.17-1985, Financial Institution Key Management (Wholesale), American Bankers Association, Washingtion, D.C., reaffirmed 1991.

50. Bellovin, S. M., and M. Merritt, Limitations of the Kerberos Authentication System, Computer Communications Review, October 1990.

51. Krajewski, M., Concept for a Smart Card Kerberos, Proceedings - The 15th National Computer Security Conference, Volume 1, October 1992.

U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161